A STUDY OF SOFTWARE MAINTENANCE COSTS OF AIR FORCE LARGE SCALE COMPUTER SYSTEMS

THESIS

Robert E. NeSmith II
Captain, USAF

AFIT/GSM/LSM/86S-15

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

2

AFIT/GSM/LSM/86

A STUDY OF SOFTWARE MAINTENANCE COSTS OF AIR FORCE LARGE SCALE COMPUTER SYSTEMS

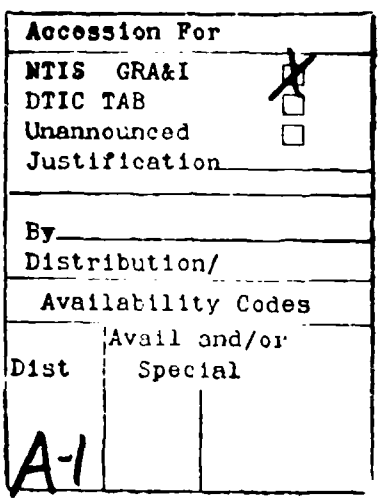
THESIS

Robert E. NeSmith II
Captain, USAF

AFIT/GSM/LSM/86S-15

Approved for public release; distribution unlimited

The contents of the document are technically accurate, and no sensitive items, detrimental ideas, or deleterious information is contained therein. Furthermore, the views expressed in the document are those of the author and do not necessarily reflect the views of the School of Systems and Logistics, the Air University, the United States Air Force, or the Department of Defense.



AFIT/GSM/LSM/86S-15

A STUDY OF SOFTWARE MAINTENANCE COSTS OF AIR FORCE LARGE SCALE COMPUTER SYSTEMS

THESIS

Presented to the Faculty of the School of Systems and Logistics
of the Air Force Institute of Technology
Air University
In Partial Fulfillment of the
Requirements for the Degree of
Master of Science in Systems Management

Robert E. NeSmith II, B.S.
Captain, USAF

September 1986

Preface

The purpose of this study was to examine software maintenance costs and its effect on Air Force life cycle costs. Only recently has work been devoted to the study of methods for maintaining software once it has been developed. Maintenance costs are rising within industry and the Air Force at an alarming rate. If efforts are not taken to reduce the current rise in maintenance, time and resources will not be available to develop new software.

I would like to thank Professor James D. Meadows, my advisor, for his support, advice, and time. Captain Roger Davis has my appreciation for providing many hours of help by reading my thesis and giving constructive comments. I would also like to thank Mr. Fred Wixon of HQ SISC/SCD at Gunter AFS AL for his help with the AFR 700-19 database. I greatly appreciate the help and support of my family during this thesis effort. Lastly, I wish to give special thanks to my wife, Barbara, for her innumerable hours of support on my behalf. She richly deserves more appreciation than is possible in words.

— Robert E. NeSmith II

Table of Contents

List of Figures

List of Tables

Abstract

Software maintenance is a growing concern throughout the software community. Due to the rising cost of computer software and the even greater increase in the software maintenance share of the budget, maintenance is becoming the major cost in a data processing organization.

This thesis examines the maintenance costs of Air Force "organic" software for the last thirty years. Generally, the cost per line of code and the total costs are rising for the large scale automatic data processing computer systems. Contractor developed software is also examined and its influence on Air Force software costs.

A STUDY OF SOFTWARE MAINTENANCE COSTS OF AIR FORCE LARGE SCALE COMPUTER SYSTEMS

1. Overview

Introduction

Industry cost studies of software have shown that of the various life cycle stages of a software system, software maintenance is the most expensive portion (25:1). Maintenance has been defined as the performance of those activities required to keep a software system operational and responsive to its users after it has been accepted and placed into production. Maintenance has also been found to consume over 50 percent of the software life cycle budget (13:65; 19:4; 14:4). Studies performed in the 1970s and 1980s have shown that the longer an error remains undetected, the more it will cost to correct that error (15:116). For example, an error found in the design stage is cheaper to correct than an error found in a later stage such as testing. Therefore, emphasis on software maintainability early in the development process can reduce costs by finding and correcting any of the various design or program errors that can occur before a system is operational (25:2). Industry estimates suggest that once a system is operational, program changes cost up to

thirty-seven times more than program changes made during development (9:51).

Further, we can expect software costs to continue to increase.

> The annual cost of computer software in the United States in 1980 was approximately 40 billion dollars or about 2 percent of the Gross National Product (GNP). Its rate of growth is considerably greater than that of the economy in general. (5:17)

The combined cost for software development and maintenance is predicted to be 13 percent of the GNP by 1990. The ratio of computer software costs to overall computer system costs has also grown tremendously. In the 1950s, 90 percent of the cost of a computer system was due to the cost of computer hardware. Currently, more than 80 percent of the system cost is attributed to the cost of software (5:18; 25:1). See Figure 1-1.

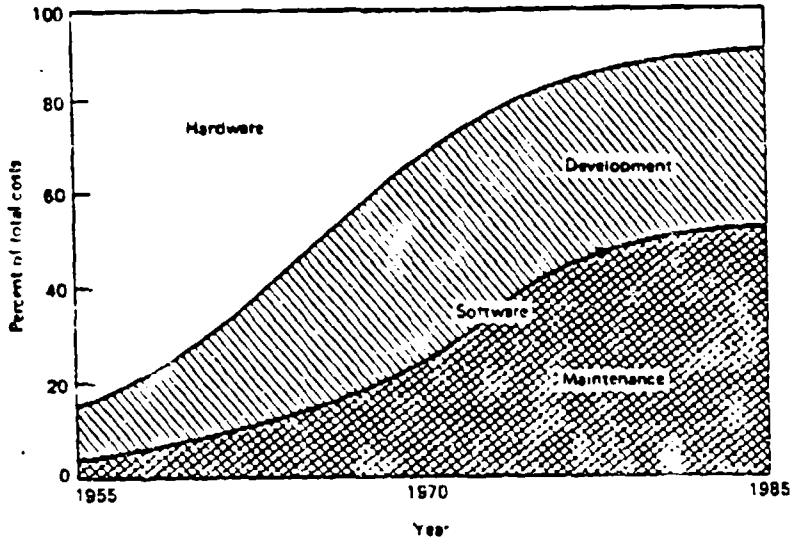


Fig. 1-1. Hardware/Software Cost Trends (5:18)

Another factor which will cause software costs to rise is the increasing reliance on software to accomplish tasks previously performed by hardware. "As software permeates virtually every defense system, cost effective and reliable software becomes increasingly urgent" (3:52). The Department of Defense (DoD) estimated in 1982 that DoD's costs for new software will increase by a factor of six to $32 billion by 1990 (3:54).

Industry tracks all efforts according to the costs incurred. Eventually all costs incurred in the operation and maintenance of software is charged to either the user or the developer depending upon the situation. If the software operates properly and produces the desired output, then the operation expense is charged to the user. If the software runs improperly, then the software goes back to the development office for repair and the costs are incurred by the developer.

However, it is difficult to determine exact software costs in the Air Force (AF). Since the Air Force usually develops software within the same unit or command, the costs all come out of the same budget. Historically, only efforts involving initial development have been tracked. These efforts included the number of lines of code written, the number of programmer hours expended, the number of months until delivery, etc. The tracking of maintenance costs was not considered a part of the software

costs. In the early 1980s, DoD instituted steps to facilitate better tracking of software costs. This was due primarily to the increased emphasis of "Fraud, Waste and Abuse" and reduced spending levels.

Finally, new and existing software systems are developed and maintained by the limited number of programmers that exist in industry and government. At the present rate of growth in maintenance requirements for existing and new systems, it is expected that all programmers will do maintenance work without additional time for developing new software after 1990 (10:74; 28:61).

These factors combined truly present a challenge to the managers of Air Force software of today and the near future. The purpose of this thesis is to analyze maintenance costs of Air Force software and discuss concepts which affect these costs.

Problem Statement

Due to the emphasis on budget reductions, ways to reduce costs become increasingly important. Industry studies show that software maintenance costs are requiring a greater share of the overall software budget. The Air Force has taken an active role in controlling software development costs but has not adequately assessed the costs of continual maintenance on its library of software used in data processing centers. Several studies have been

performed on the costs of "embedded" software for weapon systems. Few comparable studies have been completed on Air Force data processing or information systems software (20:7). An understanding of the composition of these costs is necessary before steps to reduce them can take place.

Justification

A literature search found maintenance studies performed on industry-developed software and on Air Force embedded software but not on Air Force software for large automated data processing (ADP) machines. Air Force software maintenance presents one area where significant cost reductions can be achieved to help meet reduced funding levels imposed by Congress. Methods used successfully by industry to reduce software maintenance costs may provide ways for the Air Force to reduce its software maintenance costs. The information derived from this research will aid Air Force managers in planning and assigning resources to the maintenance of Air Force software.

Objectives of Research

This thesis builds on a thesis prepared by Major Robert E. Childress, an AFIT 1985 graduate. His thesis is entitled Contractor versus Organic Maintenance for Space Command Automatic Data Processing Equipment. Although Major Childress looked primarily at manpower shortages, this thesis looks at costs. The overall

objective of this thesis is to analyze historical software costs by examining the effect of the programming language used, the number of lines of code, the magnitude of maintenance and development costs, year, contractor, and major command user. This information will allow evaluation of the following:

1. What has been the change in software development costs over time?

2. Have software development costs shown the same increase independent of the AF command or the programming language used?

3. What have been the costs of software maintenance for large data processing systems?

4. Have software maintenance costs shown the same increase independent of the AF command or the programming language used?

5. What differences are found between software development and maintenance costs?

Scope

Research of projected software requirements indicates that software needs are increasing while at the same time, budget constraints are being imposed upon DoD by Congress. Therefore, this thesis effort has potential for application in all Air Force and possibly all DoD software efforts. The information provided in this thesis

will provide the reader with an awareness of the maintenance costs within the Air Force in terms of the computer language used and the user command. It will also provide current techniques, methodology, and direction that can reduce maintenance costs for current and future software systems. With the limited AF resources available for developing and maintaining software, efforts on the part of the user command can reduce these growing costs and make better use of limited resources.

The major constraint for this project is the amount of time allowed by AFIT for thesis research. The period allows fifteen months of research work. Another constraint concerns the accounting methods used by Air Force organizations in determining software development and maintenance costs. Also, this study does not examine the subject of dedicated, system resident, or embedded software costs. Instead, only those software systems running on automated or large-scale data processing equipment by the Air Force and DoD are examined. Studies have been previously performed on the software used in embedded systems. These studies show even higher maintenance costs than on large ADP systems mentioned in this thesis. Maintenance on large ADP systems can be greater than 50 percent of the total life cycle costs or up to $2000 per line of code, whereas, embedded software maintenance costs can run as high as $4000 per line of code (26; 16:16).

II. Background

The U.S. Government is the largest user of computers in the world, with over 17,000 computers running over 5 million software packages. Most of these packages were developed at the cost of hundreds of staff years amounting to billions of dollars (4:115).

This chapter is organized into four basic sections. The first section gives background on software development and maintenance. Section two describes software life cycle costs. Section three presents some areas that increase the cost of maintenance. Finally, section four presents ways to reduce the costs of software maintenance.

Section I

Software Development. As software development costs have risen, so have the associated software maintenance costs. In the 1980s, the U.S. Government is spending on the average of $900 million annually to support software. It is predicted that this will rise more than ten-fold by the 1990s (4:115). Programmers today spend more time adapting and correcting existing software than writing new software (4:115). This means that users spend more on maintenance of software than on development of new code.

Since software development is time intensive, economical and efficient techniques to maintain existing

software must be used to reduce costs and conserve resources. Ideally, these techniques should begin when the program is developed (25:5).

During the past thirty years the number of software programs has grown at an exponential rate. Many of these software programs are still in use today. This is especially true of the government's computer systems. A recent study states that the average age of government software is ten years, three months (30:1). This is almost three times the age of software in industry (22:2). Each of these programs added to a computer system's library increases the operations and maintenance portion of the total computer facilities budget (25:5). This leaves less funds available for the development requirements (28:61).

Software Maintenance. Maintenance covers three areas: Perfective, Corrective, and Adaptive. Definitions of these terms are found in Appendix A. Perfective requires the most time from the maintenance programmer. A study by the National Bureau of Standards found the maintenance programmer spent 55 percent of his time perfecting software, 25 percent adapting, leaving only 20 percent for correction of errors (25:1).

Any maintenance work performed on a system tends to disrupt the complete program structure. Therefore, it has been found that the longer the software has been used,

the greater the chance of usage problems due to the maintenance completed on the software (11:102). Enhancements will be implemented, hardware will be upgraded, and untested "bugs" will pop up. All will require programmer work to run properly. Additionally, much of the older software is custom-made, using techniques and languages that complicate rapid problem analysis and repair (28:61).

In many cases the original writers of software have left the organization, leaving maintenance to personnel unfamiliar with the software. Documentation describing the function of the program is often out of date or nonexistent, decreasing the available reference material. This in turn increases the work load on software maintenance personnel. Some computer software maintenance is needed immediately due to the critical nature of the software. The maintenance programmer does not have the luxury of months to study the program and make any trial runs (33:22).

A National Bureau of Standards study examined the tasks involved in software maintenance. These tasks are not always performed by just one programmer or office, but the study shows the reasons why maintenance costs can add up. The breakdown of these tasks that must be accomplished and paid for is listed below (19:12).

1. Requirements Analysis
2. Design Analysis

3. User Interface
4. Design Review
5. Problem Report Recording and Control
6. Configuration Control
7. Database Modification
8. Code Modification/Recompilation
9. Code Debugging/Module Testing
10. Subsystem Testing
11. System Testing
12. Documentation Modification
13. Standards Audit
14. Code Inspections/Walkthroughs
15. Test Data Generation
16. Management Planning and Control
17. Field Delivery
18. Software Support Development/Maintenance
19. Hardware Support
20. Administrative Support

As can be seen from these tasks, maintenance is basically a microcosm of the development effort (16:18).

Despite problems with software and its use, software development has rapidly expanded and software maintenance has begun maturing as a structured science. As computer hardware has improved in cost, size, and performance, software capabilities have also improved. New

software is more powerful, easier to use, and more readily available.

Software maintenance activities are affected by software development and operations. Several software-related subjects are examined that will provide the reader with a better understanding of the role of maintenance in the total software picture.

Section II

Software Life Cycle Costs. The life cycle of a computer software system is defined as the period from its conception until it is no longer used (15:29; 24:9). The software life cycle includes six basic phases (25:2):

1. Concept and Analysis
2. Requirement Definition
3. Design
4. Code, Debug and Test
5. Installation and Evaluation
6. Operation and Maintenance

Software life cycle costs are all costs incurred in these six phases. Often software costs are erroneously thought to occur until the software is delivered to the user. As in any complex product, many development activities, such as documentation or structured program coding, affect the performance of software as well as the maintenance efforts required during the software life cycle. Any activity

involved in the development or maintenance such as documentation updates or structured coding in development of the software will affect the total life cycle costs.

Software Cost Estimation. Prior to the 1970s, little was done to allow the proper estimation of the cost of a software project. Software development was basically considered more of a nebulous form of art than a structured science. This meant that the developer did not have a firm idea of the software cost until after the project had ended. This allowed some projects to far exceed the cost considered to be the dividing point between cost-effective and cost-excessive. During the past decade, serious efforts have been made to add structure and controls to software. The most notable concept has been Top-Down Structured Programming (25:3,8).

Cost estimation techniques have followed the structured development approach. Structured cost estimation techniques generally fall into three categories: Analogy, Bottom-up and Top-Down. Each technique has its own strengths and limitations.

Analogy estimation compares programs against the cost of similar programs. The cost estimate is based on actual experience. Often, though, there is no "similar" program to compare against (15:12).

Bottom-up techniques estimate costs by estimating the costs of components or units of the system and then summing these costs. Bottom-up techniques give a detailed basis for costs, are usually accurate and stable, and can promote individual responsibility for the software. The problems with the bottom-up estimation technique are that the technique does not capture the integration costs, is time consuming, and requires a detailed knowledge of the system (15:12).

The last technique, Top-Down, is also called parametric. Top-Down estimates software system costs based on design parameters which can be partitioned among components or life cycle phases. It is fast and easy to use while requiring little detailed knowledge. At the same time, it captures the system-level costs. The weaknesses with top-down techniques are that they are less stable and not always accurate (15:12). Of the Top-Down models developed, one of the most noted of the 1980s has been the Constructive Cost Model or "Cocomo" developed by Dr. Barry Boehm of TRW (5:58). Cocomo is one of the non-proprietary models available. Several other models, each varying in the number of software factors they consider, are also in use today. A list of various models, for development and for support, is shown in Tables 2-1 and 2-2 (15:6,10).

TABLE 2-1

TOP-DOWN "DETAILED" MODELS: AVAILABLE DEVELOPMENT MODELS (15:6)

Type	Name	Acronym	Developer
Non-Proprietary	Constructive Cost Model	COCOMO	Dr. Barry Boehm
Proprietary	Programmed Review of Information for Cost Estimation-Software	PRICE-S	RCA Price Systems
	Software Life Cycle Model	SLIM	QSM Corporation (Larry Putnam)
	Jensen Systems 2	JS-2	Randall Jensen
	Freiman Analysis of Systems Technique (FAST)-Software	FAST-ES	Freiman Parametric Systems (FPS)

TABLE 2-2

TOP-DOWN "DETAILED" MODELS: AVAILABLE SUPPORT MODELS (15:10)

Type	Name	Acronym	Developer
Non-Proprietary	Avionics Software Support Cost Model	ASSCM	Syscon Corp.
	Constructive Cost Model for Maintenance (Based on COCOMO)	COCOMO-M	Dr. Barry Boehm
Proprietary	Price Software Life Cycle Model (Based on Price-S)	PRICE-SL	RCA Price Systems
	Slim-Life Cycle Option	SLIM-LC	QSM Corporation
	Fast-Software Life Cycle (Part of Fast-ES)	FAST-ES	FPS

All of the models have a common thread. Each requires information from the specific organization wishing to estimate future software costs. No one formula has been derived that can be of immediate use for all organizations (15:140-142). Some companies have the know-how and resources to produce software at a lower "per line" or "per project" cost than others. Therefore, the input factors for each company will be different.

The National Bureau of Standards reported that "software maintenance can account for up to seventy percent of each software dollar allocated" (13:65; 26). The increase in maintenance over development costs is two-fold. Once a system is placed into production its maintenance costs will take away from the total ADP budget for every year that the system is used rather than just a one-time development charge. Secondly, maintenance has a "ripple" effect on other lines of code within the same software system (11:102). Every line changed affects other lines of code, "snowballing" the changes needed. Once the changes have been made, the program has to be retested and the documentation updated (11:102; 18:4). Over time the changes cause a degradation in the quality and structure of the software, which increases the rate of future problems. All of these changes consume maintenance resources (11:102).

Section III

Maintenance Programmers. Software development projects are usually staffed with a specific mix of personnel skills to meet the requirements of the application development. People with different skills are assigned at the various phases of the development to take full advantage of their expertise. In contrast, all maintenance activities for a particular software product are often performed by one person, acting as requirements analyst, designer, coder, and tester. Although this is not always the case, the separation and assembling of a certain mix of skilled individuals is more typical during development than during maintenance (24:11).

Experienced programmers generally prefer to develop new systems rather than work on software maintenance. Therefore, inexperienced programmers usually are assigned to work in the maintenance shops to repair the defective programs. This places the responsibility for an important piece of software "on the shoulders" of someone with very little knowledge of the program and little experience in all the aspects necessary to keep it running. Often, the new programmer either is promoted out of the shop or becomes so dissatisfied from the work pressure that he quits. This places the burden of maintenance on someone else, starting the cycle all over again (23:11).

Most software systems are maintained by a different staff of programmers than those who developed the original code (13:74; 24:1,11). This is a major factor in software maintenance where the software is on average ten years old (30:1). People move on, especially in the Air Force. The usual assignment for programmers is four years. The knowledge that took years to develop about a system usually leaves with these individuals. The "corporate knowledge" in the organization concerning a software package can be a tremendous aid to the maintenance programmer in understanding a system and making the necessary changes in the software. Most large data processing organizations have their development personnel in a different section from their maintenance personnel. Industry studies have shown an annual turnover of 28 percent of the personnel in a data processing shop. The reduction of a stable programming staff negatively affects the corporate memory of the organization. The increased work load of unfamiliar software in turn can affect the morale of the remaining programmers (2:807).

All of this can lead to increased downtime on programs needing maintenance (29:5). One way to relieve some of these problems is to provide effective programming tools such as better documentation for the maintenance programmer. Training to use these tools is especially important since it not only relieves dissatisfaction but can

increase the programmer's effectiveness and efficiency. This can be critical when short time constraints are involved (2:808).

Management and User Influence. A National Bureau of Standards study identified management second only to costs as a significant area of concern in terms of software maintenance. The management problems involve:

1. Managing/recognizing software maintenance as a separate function
2. User and upper level management's perception of software maintenance
3. A lack of goals, standards, and criteria to judge performance (metrics)
4. Managing the user interface

The report concluded by stating that the significance of software maintenance was not recognized historically (24:13).

Management and the user exert significant influence in software maintenance. Software maintenance can be viewed as successive iterations of the development phases, but its uniqueness should be recognized to insure effective management (24:11). First, management must realize the difficulty of maintenance tasks. Management can influence software development and maintenance since it manages the motivation/reward system of the organization as well as

the budget, i.e., promotions and salary raises. The individuals in the programming offices in turn will plan and develop those requirements dictated by management (27:49).

Management is driven by user wants since computer support is service oriented. If the user understands that structured coding and documentation will aid in debugging software problems years from now, then the user can request management to enforce these standards. Management in turn can develop the plans needed to control the organization and its products (2:807). Therefore, training users to understand the "costs" of their requests in terms of time, money, personnel, resources, and quality of product will influence the type of requests received in the future (33:67).

Air Force Issues. Although presented from an industry viewpoint, all of these issues reflect current situations in the Air Force that affect software maintenance. In some cases, the problem is getting worse because industry programmer salares are higher than Air Force programmer salaries. Many Air Force programmers leave, placing a greater burden on the remaining Air Force personnel (9:51).

AFR 26-1, Manpower: Manpower Policies and Procedures Comparative Cost Analysis, stipulates "Positions that have direct combat support tasks under contingency or War Plans, but indirect combat support tasks in peacetime

must be identified as Military Essential." If these situations require Air Force programmers, then the only remaining resource available to the organization for additional programming or maintenance will be the contract programmer (10:64).

Section IV

Documentation. Documentation of software is another tool that has had tremendous influence on software maintenance. Software documentation is the single most effective tool for software maintenance (14:1; 32:13; 8:132,134; 1:42). Studies have shown that documentation is not being accomplished for all programs (4:115). It takes time in the development stage to write good documentation. That time and the resources involved could be used to develop additional software. The trend in both industry and government has been for management and the programming office to set documentation aside (29:5).

Since maintenance programmers depend heavily on the documentation to become familiar with a system and since maintenance costs are much greater than development costs, software documentation can be considered an investment that saves time and resources (32:13; 13:74). Automated software documentation programs are available but do not provide sufficient information for maintenance personnel. However, some documentation is better than none

as long as it is current (8:142). Documentation must be updated when software changes are made or the documentation becomes obsolete. Worse yet, unrevised documentation can be misleading and thus further increase the rate of decay and time needed for software modification (33:23).

Software Quality Assurance. Software quality assurance has received a great deal of publicity in the 1980s within industry. Software quality assurance is important in the area of maintenance because it emphasizes developing products that perform correctly when turned over to the user. This means the "corrective" errors will be reduced for the maintenance programmers. Software quality assurance begins with the premise that certain qualities are desired in a finished product. Within the Air Force, reliability and maintainability are highest on the software quality assurance priority list. Software users need to understand that specific qualities or "factors" can be built into software but only at certain costs. Some software qualities build on other qualities while others conflict and degrade (21:24). A table defining these qualities and how they affect each other can be found in Appendix G.

No one wants to develop an inferior product but quality is a concept that involves not just the programmer but everyone from the program manager to the operator.

Software quality assurance advocates that plans, metrics/ standards, and controls/reviews be implemented to insure that desired quality factors are assured (27:1,3).

Configuration Management. Configuration management is the management of change (13:149). It consists of four functions: identification, configuration control, status accounting, and auditing (24:34). It is based on a concept that software, tightly controlled, reduces the choices of inadvertent mistakes. Mistakes such as modification of software without an associated documentation update or inadequate revision testing can increase future problems for the software and, in turn, the time and resources to fix it (13:149).

In terms of testing, configuration management determines how much time and money will be spent "debugging" the errors. This is due to the error discovery cost rate. As efforts are made to find errors many will be found early, but it takes longer and longer to find the remaining errors. See Figure 2-1 (13:61).

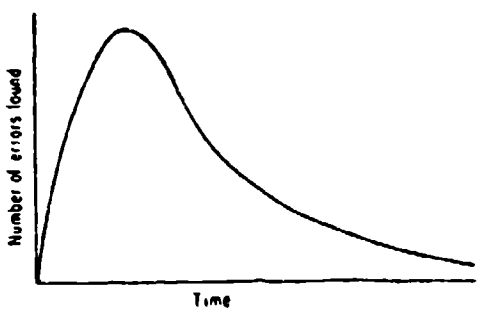


Fig. 2-1. Error Discovery Rate (13:61)

The first 95 percent of the errors may cost little in terms of dollars and time. But the last 5 percent of those errors may cost more than the first 95 percent (15:41). See Figure 2-2 (5:40).

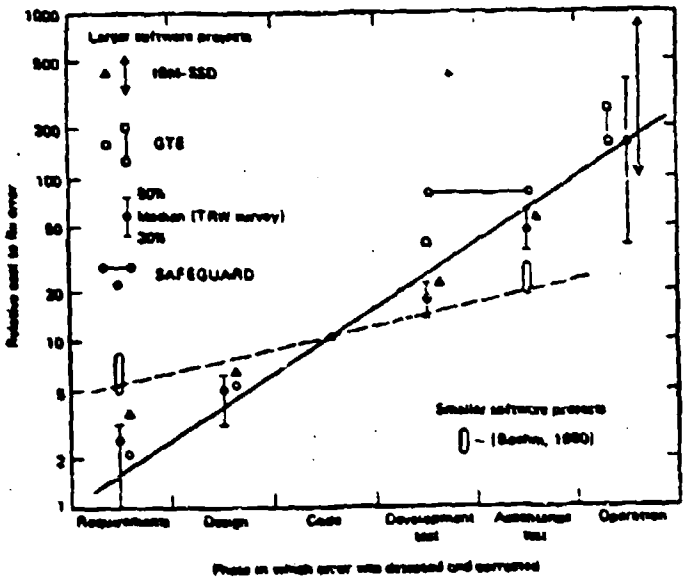


Fig. 2-2. Increase in Cost-to-Fix or Change Software Throughout Life Cycle (5:40)

A decision has to be made as to when testing stops and production begins (13:61). Programs will reach a point where they have become obsolete and further revision is useless. At this p int the software program must be rebuilt (7:1). Management and the configuration management shop must determine how to best manage the organization with its associated responsibilities and resources (31:36).

Use of software cost estimation can indicate the limit of cost effectiveness for use or rebuilding (11:101-102).

Since management determines how to utilize organizational resources, management should develop procedures for the organization to follow. This can begin with a simple plan of who does what function in information support or can develop into a highly structured organization with offices specializing in input/output control, tape libraries, software maintenance shops, etc. (24:34).

Metrics. "You cannot manage what you cannot measure" is a basic tenet behind the development of software metrics (18:1). Metrics are standards or measures that can be applied to measure the effectiveness and efficiency of software and related factors such as testing and documentation. The National Bureau of Standards has listed several areas of software maintenance metrics. Appendix I contains these lists.

Relating this to software cost estimation, Henderson and Sullivan stated that "you can't measure what you don't keep data on." Not only should the metrics be developed and used but the information derived should be archived for future reference.

Off-the-shelf Software. With the growing costs in software development and maintenance, many companies are turning to commercially available or "off-the-shelf"

software to meet specific organizational needs (17:744). This type of software can be purchased from the computer hardware manufacturer or from a third party vendor. "Off-the-shelf" software has benefits and handicaps. On the beneficial side, "off-the-shelf" software usually costs less than developing and maintaining unique software. "Off-the-shelf" software is available now and not after several months of development work. On the negative side, "off-the-shelf" software may not meet all the needs, may not be available for the organization's computer, may not have training available in its use, and may not be maintainable due to vendor policy such as restricted documentation (17:744).

III. Research Methodology

This chapter will discuss the procedures used to collect and analyze available information in order to satisfy the research objectives proposed in Chapter I. Specifically, it focuses on data collection by means of a literature review and a statistical analysis of the information contained in the Air Force Information Systems Designator (ISD) database.

AFR 700-19 Database

To adequately assess the software costs in the Air Force several methods were proposed. The first method involved contacting every command for information on costs. This was deemed inappropriate due to the time constraints of the thesis. The second method examined the Major Command Information System Plan (MISP) required by AFR 700-2 for Air Force commands. Research at the historical archives at HQ Air Force Logistics Command (AFLC) of several command's MISPs showed that maintenance costs are not recorded at this time. Maintenance costs are not required in current MISPs. The third method involved finding an established database with this information. The Data Item Designator database at Air Force Systems Command, Electronic Systems Division (AFSC/ESD) was examined. This

database covers "embedded" software rather than the ADPE software systems used in large-scale data processing computers being examined in this study. One AF database was finally found. AFR 700-19 established the Information Systems Designator (ISD) database maintained by the Standard Information Systems Center (SISC), a specialized computer center for the Air Force Communications Command. The AF ISD database was created in the 1960s to provide specific information on AF software for large automatic data processing (ADP) systems. At that time, the database was a part of the AF 300 series regulations and known as the Data Systems Designator (DSD) Database and maintained by Data System Design Office headquartered at Gunter AFS, Alabama. The purpose for the database was to share part, if not all, of the software between organizations with similar software requirements. This shared software would reduce duplication of effort and software costs. All AF commands are required to report this information for their active systems. Today, this database includes software of all types of application systems except embedded and classified. SISC at Gunter AFS, Alabama is now responsible for publishing and managing this information under Air Force Regulation 700-19.

The procedure works in the following manner. After defining a needed software capability, an organization would examine their copy of the ISD database for any

possible compatible software systems. The organization would then go directly to the owner of the functionally equivalent software listed in the database to obtain a copy. If both organizations use the software, then the second organization must also register with SISC. If nothing is found, the organization can develop the software for their use and then register it with SISC for the benefit of other organizations.

Specific Data Fields Used

SISC assigns a unique identifier called an ISD for each software system submitted to this database. The separate ISDs list primarily a description of what the software does. In addition, many fields are included in the record stating language used, hardware used, user command, development costs, maintenance costs and more. A listing of all the fields contained in this database is found in Appendix D. After studying this database for analysis, specific fields were examined for this study. These fields are:

1. YEAR--this data element lists the year the software was placed in operation.

2. CONTRACTOR--this field is either a one, two or a blank. One means the software was at least 50 percent contractor developed. Two means the software is commercially developed or "off-the-shelf" software. A blank means the software was developed within the Air Force.

3. COMMAND--this field states the primary user command.

4. LANG1--this field states the primary programming language used. In several large software systems, more than one language was used. Since the records did not break down the size or cost by language, only the first language would be used for this category.

5. LINES--this field lists the number of lines of code in the program(s).

6. DEVCOST--this data field gives the development cost of the system when it was put into operation.

7. MANCOST--this final data field lists the accumulated costs for maintenance work since the software was placed into operation. This field is updated yearly although no field indicated the last year of update.

Computer Support

The ISD database is designed to be a report generator. The records and fields are variable length. SISC does not perform any statistical analysis on this data; it is used only to give a description of that ISD record. Since this study examined the size and costs of the software, verification of the costs was desired. The data is required to be sent to SISC from the commands and changes are to be updated yearly. Although SISD verified that the data sent for this study was the same as their

database, all attempts to verify this data from a second source were unsuccessful. It cannot be assumed that the data is accurate neither can it be assumed to be in error. Therefore, the results of this study are only as good as the data available.

In order to accomplish statistical analysis, another database was built on the AFIT Classroom Support Computer (CSC) using the information extracted from the ISD database. The CSC is a Digital Equipment Corporation VAX computer with the Virtual Memory Storage (VMS) operating system. This computer system was used because of the availability of the "SAS" statistical package. "SAS" is a copyrighted statistical package that allows data to be analyzed and reported in several different ways in the same computer run.

One of the first steps in the analysis required a frequency count of the number of occurrences of the programming languages, the various commands, and the number of programs developed under contract or off-the-shelf software. It was decided that to be a representative sample, thirty examples of the field being examined must be present. A copy of the statistical program is found in Appendix B.

Research Objective One

The first objective, what has been the change in AF development costs over time, will be answered by analyzing the difference in the cost per line of code over time. After being examined as an AF total per year, the data will be analyzed according to the programming language used, by the AF user command and, finally, contractor software development costs. Examining the total Air Force development costs, 1342 examples were available from the years 1955 through 1986. The number of lines of code developed during a specific year was divided into the total development costs for that year.

Total Development Cost/Total Number of Lines
= Cost per Line

Most of the data was studied by year. If a record did not list a specific year in the year field, that record was deleted from further analysis in the development cost, since it could not be placed into a specific year category. This resulted in 254 records being deleted from further analysis.

Air Force commands were required to report the development and maintenance costs with their ISD information. If actual costs were not available, AFR 700-19 provided a formula for estimating the costs of $20 per line of code for development costs and $80 per line of code for

maintenance costs. There is no indication in the records of whether the actual costs or the AFR 700-19 formulated costs were reported to SISC. Although some of the ISD records may list costs above the actual costs, just as likely some costs would be below the actual costs. Averaged across the 1600+ records, it is likely that the overall costs per line would be close to the true actual cost of AF software per line.

The average cost per line of code for each command is generated by the total cost for that command divided by the total number of lines of those systems. This same procedure will be performed for each programming language used to give the average AF cost by year per line of code for that language.

Research Objective Two

The second objective, have these development costs shown the same increase independent of the AF command or the programming language used, will build on the first objective. Where objective one looked at actual numbers and cost per line, objective two will look at total dollar trends. SAS will be used to build charts showing development costs compared over time. The year-by-year progression will be shown on the same chart. Charts showing combined AF, unique programming language, and specific commands will be built. The detailed numbers will be

available for in-depth comparison of the cost ratios compared to lines of code.

Research Objective Three

Objective three, what have been the costs of software maintenance for these systems, is similar to objective one except that maintenance costs will be examined instead of development costs. Again, the categories will be the cost per line of maintenance work as a whole within the AF, then the programming language used, the user commands and, finally, contracted software. The analysis will then proceed year by year to look for any indication of rising or lowering at an unusual rate.

Total Accumulated Maintenance Costs/Total Number of Lines = Cost per Line

If any year showed unusual figures, a table listing the specific category and year will provide further information to clarify the peculiarities.

From this analysis, the cost of Air Force software maintenance should be found and a comparison against contractor costs can show which is the most economical. Since the industry average costs are already known from the literature search, analysis of the Air Force costs can also provide a comparison in several ways, between the Air Force versus industry and contracting versus industry.

As with the first and second objectives, the frequency count will be used to decide on the specific languages and commands to examine. For additional comparison in the event of unusual findings, all remaining languages will be placed in an "other" language category and the remaining commands will be placed in an "other" command category for analysis and comparison against the specific languages and commands.

Research Objective Four

Objective four, have maintenance costs shown the same increase independent of the AF command or programming language used, will be examined in a similar fashion to objective two. Here, the purpose is not to look at the efficiency or inefficiency in the maintenance costs, but to look at the total volume effort--the quantity. The total maintenance effort will be examined within the Air Force year by year. This will be sorted by languages for analysis then resorted and analyzed by commands to see if any particular unusual costs are found. Finally, the contractor software will be analyzed as a whole and by year. This information, placed on a graph, should show total costs and trends in maintenance cost.

Research Objective Five

The final objective, what differences are shown between development and maintenance costs, analyzed as

were the previous objectives, will show whether Air Force maintenance costs are greater than the development costs for the same systems. If this proves to be the case, then this would show a trend similar to that in industry. The literature search found industry software maintenance costing at least as much as the development costs. The Air Force costs for development and maintenance may not equal industry's, but if the percentage differences between Air Force development to maintenance is similar to industry's development to maintenance ratio then the same problems occurring in industry may be occurring in the Air Force.

If the analysis of the data shows that the Air Force software maintenance is similar to industry's then the methods, techniques, and problems in industry may be the same as the Air Force's. Improvements in industry may be used to reduce the maintenance costs in the Air Force well.

IV. Research Observations

To better understand the costs of Air Force software, analysis of the cost and size data from a computer software database was categorized and totaled using the SAS statistical package. The details and analysis are presented here.

To begin the analysis of the database, the SAS frequency option was used to categorize the data. The frequency option gave a count for each occurrence in each field. Table 4-1 shows the different commands represented with their respective frequency count. Thirteen commands contain enough samples to be selected for individual analysis. The remaining commands were grouped into an "other" command category for comparison. The thirteen commands are Air Force Accounting and Finance Center (AFAFC), Air Force Communications Command (AFCC), Air Force Logistics Command (AFLC), Air Force Systems Command (AFSC), Air Training Command (ATC), Air University (AU), Data Systems Design Organization (DSDO), Electronics Security Command (ESC), Military Airlift Command (MAC), Pacific Air Forces (PACAF), Strategic Air Command (SAC), Tactical Air Command (TAC), and United States Air Forces Europe (USAFE).

TABLE 4-1

COMMAND FREQUENCY COUNT

Command	Frequency	Command	Frequency
AAC	11	ANGSC	1
AFAA	1	ATC	68
AFAFC	27	AU	48
AFCAC	1	DCA	13
AFCC	240	DLA	1
AFCOMS	2	DNA	1
AFCOS	1	DSDO	30
AFDSDO	1	ESC	49
AFESC	1	JCS	2
AFIS	16	JDA	1
AFISC	5	JDSSC	4
AFLC	394	MAC	153
AFLMC	6	MMSSA	1
AFMEA	1	PACAF	62
AFMPC	6	SAC	130
AFMSC	11	SPACECMD	20
AFMSMET	1	TAC	92
AFOTEC	2	TACC	2
AFRES	4	TPSC	21
AFSC	96	USAF	5
AFWL	1	USAFA	5
ANG	1	USAFE	63

Table 4-2 shows the different languages and their frequency count. The languages were handled in a similar fashion to the commands. Three languages, Assembler, FORTRAN, and COBOL, had enough samples for each individual study. The remaining programming languages were grouped into an "other" language category.

TABLE 4-2

LANGUAGE FREQUENCY COUNT

LANG1	Frequency	LANG1	Frequency
AFOLDS	5	GMAP	22
ALGOL	3	JOVIAL	10
APL	3	PASCAL	1
ASSEMBLER	54	PL/1	5
BASIC	20	RAMIS	1
BASIS	1	RPG	3
COBOL	1185	SIMSCRIPT	6
DAD	3	SIS	1
DBMS	4	TURN-KEY	1
FORTRAN	197	UTILITIES	58
		UTILITY	1

Research Objective One

What has been the increase in development costs over time?

To compare these costs, two additional standards were developed in addition to the industry average mentioned in Chapter II. The total AF cost per line was calculated from the total number of lines and total development costs. The second standard was developed for each

language or command. In both cases the "years developed" was ignored. This table is shown in Table 4-3. Developing these standards using the total records regardless of year is important to the database because several hundred records lacked "year developed" preventing these records from being categorized for analysis. With the total records for development costs calculated, the records could be categorized and compared against these three standards (industry, AF, category) over time. Tables were built for the languages over time and commands over time. These can be viewed in Appendix E. Graphs showing the cost per line changes over the various years can be found in Appendix C.

AF Total Cost. Examining the cost per line of code, the AF developed cost per line of code is found to fall below the industry cost range of $50 to $200. This cost was calculated by dividing the total development cost by the total number of lines.

$1,241,355,956 / 653,760,528

= $1.90 per Line of Code

The calculations on the cost per line year by year showed the cost in the high teens until 1972, at which time it dropped into single digits. One exception occurred in 1962 with $233.60 per line of code. Examining this exception, it was found that the total development cost was

TABLE 4-3

DEVELOPMENT COST PER LINE BY CATEGORY

	No. Usable Records	Total Cost per Line of Code	No. Years > AF Yearly Average Cost	No. Years = AF Yearly Average Cost	No. Years < AF Yearly Average Cost	No. Years ≥ Industry Cost Range
Language						
ASSEMBLER	58	2.70	6	0	6	0
COBOL	1185	11.08	10	10	11	0
FORTRAN	166	.47	16	1	5	3
OTHER	128	5.23	9	2	5	3
Command						
AFAFC	-	16.71	11	0	1	1
AFCC	-	.31	9	2	14	0
AFLC	-	11.61	15	3	9	0
AFSC	-	19.52	10	1	9	17
ATC	-	1.77	5	0	14	0
AU	-	2.77	1	1	14	0
HQ SISC	-	19.41	4	0	1	1
ESC	-	.68	N/A	N/A	N/A	0
MAC	-	16.65	16	2	3	0
PACAF	-	1.67	0	0	4	0
SAC	-	16.65	10	3	7	1
TAC	-	17.51	2	3	8	1
USAFE	-	5.03	7	0	5	0
OTHER	-	6.68	12	0	9	0
Contractor Developed	56	29.25	8	3	4	2

not high compared to the other years but the number of lines of code was unusually low which boosted the price per line. Although SISC verified their data, this cost was caused by a large HQ MAC development cost. The other two commands that showed development costs in 1962, AFCC and AFLC, averaged $20 and $18.68 respectively. This means that excluding the HQ MAC software system, 1962 followed the average with the other years, 1955 to 1971.

Programming Languages. Examining the languages next, Assembler, COBOL, and FORTRAN had enough examples to list each as a unique category. All other languages were grouped together into an "Other" category. Refer to Appendices C and E for the related graphs and tables.

The Assembler category had fifty-eight records through 1985 and only eight of the fifty-eight assembler programs listed as operational before 1975. The total Assembler cost per line of code was $2.70. Generally, the cost per line was higher than the Air Force average of $1.90 per line of code.

The COBOL language was the largest category, with 1185 records, of the four language categories observed. Dividing the total development cost by the total number of lines gave an average development cost per line of $11.08. The development cost per line for years 1955 to 1961 rose above $20 per line. Looking year by

year for thirty-one years, the costs never rose above $34.98 per line. Again, this is below the $50 to $200 per line range in industry.

The FORTRAN programming language had 197 records with a total cost per line average of $.47. Thirty-one records were removed from further analysis because of insufficient data in the year field. The 166 remaining records covered the period from 1962 until 1984. Comparing against the total AF yearly average, programs written in FORTRAN were higher overall than the yearly average. A major factor for the low overall development cost per line is due to one system written in FORTRAN in 1972 by AFCC. This command had a system containing 500,000,000 lines of code at a cost of $100,000,000. If this system is removed from the calculations, the FORTRAN cost per line rises to $15.95.

The final category examined is the "Other" category. Here all the languages that did not contain at least thirty representative systems were grouped together for analysis.

The "Other" category contained 128 systems with a total cost per line of $5.23. This is almost three times higher than the AF total average of $1.90 per line. Comparing year by year, the "Other" category had higher overall costs.

Command. The development costs were also examined by command. Thirteen commands contained enough systems to be examined separately and the remaining thirty-one representative commands were placed in the "Other" category for a combined analysis. Refer to Appendices C and E for the graphs and tables.

The first command examined, AFAFC with twenty-seven examples, had a total average cost per line of $16.71 which is above the total Air Force average.

AFCC had 240 systems registered in the database. The average development cost per line was $.31. This was the lowest command total development cost per line and far below both the AF and industry cost per line. A major reason for the low command total cost per line is the 500,000,000 line system developed in 1972 at a cost of $100,000,000 and a 50,000,000 line system developed for $120,000 in 1975. These two systems had a major effect not only on AFCC but on the AF yearly and total cost per line. Twenty-five years were represented and during most of those years, AFCC was below the AF yearly cost per line.

AFLC's file contained 394 systems which was the most for any command. Only thirteen of AFLC's systems were deleted because of an incomplete year field. AFLC had a total average of $11.61 per line which is well above the AF total average. Examining each year found that AFLC exceeded the AF yearly average fifteen times, was below

the AF yearly average most years, but was always below industry costs.

AFSC had over 2.5 million lines of code in ninety-six systems giving an average cost of $19.52 per line for development. During a twenty-year period, ninety-two systems were listed as active.

ATC excluded only three of its sixty-eight systems due to a mission year field. The total average cost per line was $1.77 which is slightly below the AF average cost per line. ATC costs year by year were overall below the AF average.

AU had forty-eight systems with an average cost of $2.77 per line which is slightly higher than the AF average. Only thirty of the systems indicated a specific year with a time span of sixteen years. AU's year-by-year costs were below the AF yearly averages.

DSDO, now known as SISC, had thirty systems listed covering a five-year period. The total average per line of code cost is $19.41 or ten times greater than the AF total per line cost. Only five of the thirty systems listed a year in the required field. SISC year-by-year averages also exceeded the AF year-by-year averages.

The next command examined was ESC with forty-nine computer software systems with a low total per line average cost of $.68. Only thirty-eight of the systems could be analyzed by year due to missing data in the year field.

ESC's systems are relatively new suggesting that the cost would be higher than earlier systems developed elsewhere. This was not the case. Every year lists a lower yearly average per line cost, except 1982 with $5.37, than the AF total average per line cost.

MAC had the third largest sample with 153 systems averaging $16.65 per line of code. This is almost nine times greater than the AF average cost. Only five MAC systems lacked a year identifier with the remaining systems covering a twenty-year period. Of these twenty years analyzed, most were above the AF total. The yearly average cost per line was skewed in 1967 due to a large development cost on one system. If that system is removed from the calculations, the per line cost for that year is reduced to $20. The same situation occurs again in 1970 with similar results when changed. Overall, the per line costs are well within the range of AF costs, although they are below the average in all cases for industry.

PACAF was unusual to analyze because it had sixty-two systems registered but only twenty-two of them listed the year developed. Of these twenty-two, fifteen were lacking development cost and two lacked the number of lines in the system. With the remaining information for the command, the total average development cost per line was calculated to be $1.67. This is slightly below the AF total average cost per line. Every one of the four years

available for comparison were below the AF yearly average cost per line. All four were below three dollars per line.

SAC was examined next. The 130 systems covered 1963 through 1985, although twenty-two systems lacked appropriate information to categorize by year. SAC's total average cost per line is $16.65. This is nearly nine times greater than the equivalent AF cost per line. For the twenty years examined, SAC's cost per line was above the AF average yearly cost most times. Only one year, 1972, fell in the same range as industry's.

TAC had ninety-four systems listed which averaged $17.51 per line. This is over nine times greater than the AF average. Only seventy-one of these systems listed the year developed covering the thirteen-year period. These thirteen years were mostly above the average.

USAFE is the final unique command examined. It contained sixty-three records which gave a total average cost per line of $5.03. Only fifty-six records had the year indicated. These records covered twelve years. The USAFE yearly total cost per line exceeded the AF yearly total cost per line about half the time.

Finally, the remaining commands were grouped together under a heading of "Other." This added up to 126 records covering twenty-nine commands. Their combined records gave a total average per line cost of $8.68, over four and one-half times larger than the AF total average

cost per line. The records covered a total of twenty-one years from 1965 until 1985. During this period, the "Other" category exceeded the AF yearly average cost per line about half the time.

Contractor Developed Software. Examining the costs of development for contractor-developed software, sixty-seven records in the database were at least 50 percent contractor developed. They give a total average development cost per line of $29.25 which is almost seventeen times the AF equivalent. However, this is below industry average costs of $50 to $200 per line of code. It would be expected that contractors would account for their costs more along the lines of industry than the AF.

Research Objective Two

Have these development costs shown the same increase independent of the AF command of the programming language used?

To examine the rise in software development costs, the costs were graphed by year and placed against a standard y-axis of $200,000,000. Looking at the combined AF graph shows a cyclical but steadily rising software development cost.

Programming Language. Looking at the various languages, Assember did not really show enough of a

pattern to draw any conclusions. See Appendix F. The chart indicates that assembler costs are declining. This may follow with the emphasis towards use of higher order languages in computer systems.

COBOL systems were well-represented and showed the same pattern as the AF chart, cyclical but steadily rising. FORTRAN indicated just the opposite. Costs are cyclical but diminishing which may indicate either reduced cost per line of FORTRAN, smaller programs, or less use of FORTRAN for system development. The "Other" category shows only a small period which may not be a full cycle. These costs appear to be diminishing as well. See Appendix F.

Command. The commands show more of a continuous flow in development costs rather than a cyclical rise. AFCC, SAC, and TAC do show occasional peaks but most, like AFLC, indicate a budget with computer systems being developed when they can be fitted into that budget. Air Force organizations have requested increasing software support over the past thirty years. Budgets were increased accordingly which allowed ADP offices to provide the resources needed. Large projects though were multi-year development projects and the way the costs were recorded only showed up in the AFR 700-19 database when the computer systems were placed into operation. This in turn showed up as peaks on the graphs.

Research Objective Three

What have been the costs of software maintenance for these systems?

As with research objective one, objective three began comparisons by first developing additional standards. The total AF cost per line was calculated from the total number of lines and total maintenance costs. The second standard was developed for each language or command. In both cases, the "years developed" field was ignored. See Table 4-4. This is important to the database because several hundred records lacked "year developed" preventing these records from being categorized for analysis. Now the records could be categorized and compared against these three standards (industry, AF, category) over time. Specific tables were built for the languages over time and commands over time. Refer to Appendices C and E.

AF Total Cost. Examining the cost per line of code, the AF accumulated maintenance cost is found to be much lower than industry averages of $50 to $2000. The total number of lines of code written was 653,760,528. The total maintenance costs for these lines was $1,489,294,836 or $2.27 per line of code.

Programming Languages. Examining the languages next, Assembler, COBOL, and FORTRAN had enough examples to list each as a unique category. All other languages

TABLE 4-4

MAINTENANCE COST PER LINE BY CATEGORY

	Total Cost per Line of Code	No. Years > AF Yearly Average Cost	No. Years = AF Yearly Average Cost	No. Years < AF Yearly Average Cost	No. Years ≥ Industry Cost Range
Language					
ASSEMBLER	2.08	4	1	6	0
COBOL	22.24	10	10	11	0
FORTRAN	.25	13	0	9	3
OTHER	2.99	5	0	5	4
Command					
AFAFC	29.84	8	0	4	1
AFCC	.21	9	5	11	2
AFLC	17.75	17	4	5	8
AFSC	21.82	11	1	8	2
ATC	4.10	2	3	14	0
AU	4.25	2	1	13	0
HQ SISC	70.14	4	0	1	2
ESC	.17	N/A	N/A	N/A	0
MAC	39.40	13	0	7	9
PACAF	.24	4	0	0	0
SAC	27.56	12	0	8	7
TAC	6.14	2	3	9	1
USAFE	1.36	2	0	10	0
OTHER	15.76	13	2	7	4
Contractor Maintenance	12.70	6	3	6	2

were grouped together into an "Other" category. The graphs and tables are found in Appendices C and E respectively.

The Assembler category had fifty-eight records through 1985 with only eight of the fifty-eight assembler programs listed as operational before 1975. The total Assembler cost per line of code was $2.08. Generally, the accumulated maintenance cost per line was lower than the Air Force average of $2.27 per line of code.

The COBOL language was the largest category of the four language categories observed with 1185 records. Dividing the total accumulated maintenance costs by the total number of lines gave an average maintenance cost per line of $22.24.

The FORTRAN programming language had 197 records with a total per line average of $.25. Thirty-one records were removed from further analysis because of insufficient data in the year field. The 166 remaining records covered the period from 1962 until 1984. Comparing against the total AF yearly average, programs written in FORTRAN were higher than the yearly average thirteen out of twenty-two years and lower than the average nine years.

The final category examined is the "Other" category. Here all the languages that did not contain at least twenty-seven representative systems were grouped together for analysis. A listing of the various languages can be found in the frequency table in Table 4-2 (page 40).

The category "utilities" or "utility" actually contained many different languages so these were placed into the "Other" category despite having fifty-nine representative systems. The "Other" category contained 128 systems with a total cost per line of $2.99. This is only $.80 higher than the AF total average of $2.19 per line.

Command. The maintenance costs were also examined by command. Thirteen commands contained enough systems to be examined separately and the remaining thirty-two representative commands were placed in the "Other" category for a combined analysis. Refer to Appendices C and E for the charts.

The first command examined, AFAFC with twenty-seven examples, had a total average cost per line of $29.84 which is well above the total AF average of $2.19. During the twelve years examined, AFAFC exceeded the yearly AF average maintenance cost per line for eight of those years and the remaining years were below the AF yearly average.

AFCC had 240 systems registered in the database. The average maintenance cost per line was $.21. This was the lowest command total accumulated maintenance cost per line and far below both the AF and industry cost per line.

AFLC's file contained 394 systems which was the most for any command. Only thirteen of AFLC's systems were deleted because of an incomplete year field. AFLC had a

total average of $17.75 per line which is well above the AF total maintenance average of $2.19.

AFSC was the next command examined. This command had over 2.5 million lines of code in ninety-six systems giving an average maintenance cost of $21.82 per line. During a twenty-year period, ninety-two systems were listed as active. The AF yearly average cost per line was exceeded about half the time.

ATC excluded only three of its sixty-eight systems due to faulty year fields. The total average cost per line was $4.10 which is almost double the AF average maintenance cost per line. ATC was below the AF average most of the time.

AU had forty-eight systems with an average maintenance cost of $4.25 per line which is almost double the AF average. Only thirty of the systems indicated a specific year with a time span of sixteen years. Most years were below the AF yearly total average.

SISC had thirty systems listed covering a five-year period. The total average per line of code cost is $70.14 or thirty-two times greater than the AF total per line cost. Only five of the thirty systems listed a year in the required field. Four of the five years were above the AF yearly total average.

The next command examined is ESC with forty-nine computer software systems with a low total per line average

cost of $.17. Only thirty-eight of the systems could be analyzed by year due to missing data in the year field. ESC's systems are relatively new suggesting that the cost would be lower than earlier systems developed elsewhere. This appears to be the case. Every year lists a lower yearly average per line cost than the AF total average per line cost.

MAC had the third most systems registered with 153 systems averaging $39.40 per line of code. This is almost eighteen times greater than the AF average cost. Only five MAC systems lacked a year identifier with the remaining systems covering a twenty-year period. Of these twenty years analyzed, thirteen were above the AF yearly total average cost. The yearly average cost per line was skewed in 1967 due to a large maintenance cost on one system that did not have a listing for the number of lines of code. As found in the development analysis, the other system for that year used its lines of code figure in the calculations for both systems. If the first system is removed from the calculations, the per line cost is reduced to $4.70. The same situation occurs again in 1970. By deleting the computer system without the number of lines of code, the maintenance cost per line is reduced from $2107.43 to $1120. Overall, the per line costs are well above the range of AF costs and within the range for industry averages.

PACAF was unusual to analyze because it had sixty-two systems registered but only twenty-two of them listed the year developed. Of these twenty-two, sixteen were lacking maintenance costs and two lacked the number of lines in the system. With the remaining information for the command, the total average maintenance cost per line was calculated to be $.24. This is below the AF total average cost per line. Every one of the four years available for comparison was below the AF yearly average cost per line. All four were below three dollars per line. However, since these systems are relatively new, maintenance costs may not have had time to accumulate.

SAC was examined next. The 130 systems covered 1963 through 1985, although twenty-two systems lacked appropriate information to categorize by year. SAC's total average cost per line is $27.56. This is nearly thirteen times greater than the equivalent AF cost per line. For the twenty years examined, SAC's cost per line was above the AF average yearly cost twelve times. Seven years, SAC's per line cost were in the same range as industry's. Surprisingly, in the year 1982, SAC had already accumulated enough maintenance costs to boost the per line cost to $716.57. This was the highest maintenance cost in 1982 of any of the commands.

TAC had ninety-four systems listed which averaged a per line cost of $6.14. This is almost three times

greater than the AF average. Only seventy-one of these systems listed the year developed covering a fourteen-year period. Of these fourteen, nine of the years were less than the AF yearly average.

USAFE is the final unique command examined. It contained sixty-three records which gave a total average cost per line of $1.36. Only fifty-six records had the year indicated. These records covered twelve years. The USAFE yearly total cost per line for ten years was below the AF yearly total cost per line.

The remaining commands were grouped together under a heading of "Other." This added up to 126 records covering twenty-nine commands. Their combined records gave a total average per line cost of $15.76, almost seven times larger than the AF total average cost per line. The records covered a total of twenty-one years from 1965 until 1985. During this period the "Other" category exceeded the AF yearly average cost per line thirteen times.

Contractor Maintenance Costs. Finally, examining contractor maintenance costs shows a total average maintenance cost of $12.70. This is almost six times greater than the AF total maintenance cost but far below the industry average. Most of these systems became operational in 1981 or later which may explain the lesser amount. Maintenance costs have not been able to accumulate for these

systems. The year-by-year analysis showed that contractor maintenance costs exceeded AF maintenance cost six years, were equivalent three years and below the remaining six years. These were below the industry averages in all but two years, 1965 and 1966, which may be a reflection of the extreme age of the systems and the continued accumulation of maintenance costs.

Research Objective Four

Have these maintenance costs shown the same increases independent of the AF command or the programming language used?

The data was placed on graphs and examined for any patterns. Initially it was thought that maintenance costs would reflect a decrease over time; that is, a steadily decreasing line from 1955 to 1986. The logic behind this was that the oldest system would have accumulated the most costs in the maintenance category. This is not the case as can be seen in the total AF graph in Appendix F. The explanation appears to be that the oldest systems tend to be phased out leaving only a few systems to graph. At the same time, the newer systems being more numerous would show a greater combined maintenance cost.

Another way of showing this would be to take the total maintenance cost for that year and divide that number by the difference between 1986 and the year

developed. This gives the average amount spent each year for the systems developed during a specific year. Performing this on every year in the total AF maintenance cost table shows a cyclical but steadily rising amount.

This method also shows how much on average that the AF will spend this year, 1986, for all the systems listed in AFR 700-19. If none of the systems is removed from inventory, the combined amount is $236,339,850 spent on maintenance for all the active systems listed.

It was noted in this analysis that maintenance costs for these systems generally exceeded the development costs. This supports the industry claim that maintenance costs are the greater part of the total life cycle costs. These systems are still operational so they will continue to accumulate maintenance costs, raising the percentage over time. Looking at the year-by-year total AF costs, twenty-one of the thirty-one years examined had greater maintenance costs than development costs. See Appendix F.

Programming Language. Examining the data further, the costs recorded for systems developed in Assembler showed total development costs equivalent to total maintenance costs. The year-by-year analysis showed maintenance above development four years, below six years and equal one year. COBOL showed 66 percent of the total life cycle costs to be maintenance. The year-by-year analysis showed

twenty-six of thirty-one years to have greater maintenance costs than development costs. Systems written in FORTRAN showed greater development costs than maintenance costs. Of the twenty-three years listed, twelve years had greater maintenance costs. These statements seem contradictory; however, closer examination shows that the FORTRAN systems had several extremely large development costs. This skewed the comparison, overall, but not in the year-by-year analysis. The "Other" category showed development costs greater than maintenance costs overall and for eight of the fifteen years on record. See Appendix F.

Command. The individual commands showed the same situation. AFAFC had greater maintenance costs than development costs and for eight of the twelve years. AFCC had lower total maintenance costs. In fourteen of the twenty-five years examined, the maintenance costs were equal to or greater than development costs. AFLC showed greater total maintenance costs and for yearly totals in twenty-one of the twenty-seven years of data. AFSC had greater total maintenance costs and yearly totals for thirteen of twenty years. ATC showed almost twice the total maintenance costs to total development costs. In thirteen of nineteen years, maintenance was either equivalent or greater than development. AU showed total maintenance costs to be 65 percent of the total life cycle

costs. Of the sixteen years, eleven years had equivalent or greater maintenance costs. SISC, with only a small sample to use, showed maintenance accounting for 77 percent of the total life cycle costs. Four out of the five years also showed greater maintenance costs. ESC with its relatively new systems showed greater development costs both for the totals and in three out of six years. MAC followed the majority of the commands with greater total maintenance costs and for yearly costs for seventeen of twenty-one years. PACAF had much lower maintenance to development costs for both the totals and for every year. SAC's data shows greater maintenance costs for the totals and for eleven of twenty years. TAC showed greater total development costs. Ten of the fourteen years showed a greater development costs for TAC. The final command examined, USAFE, had lower total maintenance costs overall and for every year examined. The "Other" category had greater total maintenance costs overall and for fifteen of twenty-one years. See Appendix F.

Research Objective Five

What differences are found between the development and maintenance costs?

Combining the information of research objectives one with three and research objectives two with four showed that maintenance costs are indeed greater than

development costs in most cases despite the command or programming language. See Appendix E for the tables and Appendix F for the graphs.

V. Conclusions and Recommendations

Conclusions

The purpose of this thesis was to examine software maintenance costs in the Air Force on large scale computer systems. With the continued dependence upon computers, software costs have escalated at an alarming rate. Of major concern is the even greater increase in the cost of software maintenance. This increase has demanded an increasing amount of limited Air Force resources for computer software.

This study has shown that Air Force software maintenance costs have risen as have all industry-wide software maintenance costs. They now account for over 50 percent of the software life cycle costs. Generally, the older a computer system the more maintenance it requires. Therefore, the Air Force can expect to see a continuing rise in its software maintenance costs. The average age of Air Force software is ten years three months but some are as old as thirty years.

Air Force and AF contractor maintenance costs are generally less expensive than costs for the software industry as a whole. Contractor costs are still larger overall, than maintenance costs for AF developed systems. Despite this, contractor maintenance programmers have some

advantages over AF programmers. They have developed long-term familiarity, experience, and system skills that are unavailable to AF personnel. Air Force personnel, with three to four years per duty station, do not have adequate time to develop those skills in maintaining computer systems that are ten to thirty years old. Therefore, contractor software will continue to be an important factor in future software maintenance.

Research Objective One. What has been the increase in development costs over time? The information in this study has not shown any appreciable rise in software development costs over the last thirty years. In fact, the Air Force-wide development cost per line in 1955 was $20 per line. The same cost in 1985 was only $4.11. The costs during this thirty-year period fluctuated but the last six years were all under $10.

Research Objective Two. Have these development costs shown the same increase independent of the AF command or the programming language used? From 1955 to 1985, the total Air Force development costs rose from $2,661,500 to $63,037,789. Over this thirty-year period the costs had a cyclical but steadily rising amount. Assembler and FORTRAN showed a decrease in use while COBOL and the "Other" languages have shown large increases. The various commands listed showed cyclical but steady increases in their

software development costs except for MAC, PACAF, SAC, and TAC. The use of contractor developed software also showed a steady rise from $1,553,100 in 1955 to $14,808,317 in 1984.

Research Objective Three. What have been the costs of software maintenance for these systems? In general, the analysis of the data has not shown any rise in maintenance cost per line. The analysis of the data has shown a decrease in the Air Force cost per line from $82.10 in 1955 to $5.45 in 1985. Assembler, COBOL, and FORTRAN showed total decreases in the maintenance cost, although these costs showed a one dollar increase from $4.80 in 1964 to $5.80 in 1984. Contractor costs went down from $64 in 1965 to $17.44 in 1984. All of these costs were well below the industry's average of $2,000 per line of code.

Research Objective Four. Have these maintenance costs shown the same increase independent of the AF command or the programming language used? This area of analysis showed that the oldest systems have accumulated some large maintenance expenses over the years. The rate at which these costs are increasing is also rising over time. Where some software systems required thirty years to accumulate maintenance costs equal to the development costs, software systems developed as late as 1985 have already accumulated maintenance costs equal to or greater than the development costs. The accumulated maintenance costs for 1955 were

$10,926,057 for thirty years while the systems developed in 1985 have already accumulated $83,642,709 in maintenance costs. All four language categories showed cyclical but steady increases in the rate at which systems accumulated maintenance costs. Also shown for the four languages was a greater rate of growth in maintenance costs for the newer systems than for the older systems.

An analysis of maintenance costs of the four categories; Air Force, languages, commands, and contractor, revealed two things. First, commands showed strong cyclical tendencies but revealed no significant findings. Second, maintenance costs on contractor developed software increased at a lower rate than maintenance costs on Air Force developed software. The money spent each year for software maintenance for thirty years worth of software, averages out to $23,339,850 per year. This includes systems operational in 1986 that fell into this thirty-year period.

Research Objective Five. What differences are found between the development and maintenance costs? This thesis has shown the maintenance cost rising at a greater rate than development. Just as software costs in industry, maintenance costs have become greater than development costs. This means that maintenance, which is a continual cost, will take a larger and larger share of the

organization's software budget. This was true not only for the different languages but for most of the commands as well, except for ESC, TAC, and USAFE.

The total maintenance costs for contractor developed software were lower than the development costs. This may indicate that the software did not require as much maintenance as Air Force developed software.

Limitations of Research

The reader should understand that the actual figures given in this database for development and maintenance costs appear very low compared with industry costs. Verification of the data during the research proved impossible. Without this verification, several questions were raised. First, if the computer system's costs show figures similar to the $20 and $80 development and maintenance estimates respectively, does this mean that the organization did not record their costs? AFR 700-19 stated in the directions to multiply $20 by the number of lines of code in the system to give the estimated development cost. The maintenance cost was derived by multiplying $80 by the number of lines in the system. Second, if the organization used the maintenance cost formula from AFR 700-19, did the organization multiply the cost by the total lines of code or just the lines of code changed?

Contractor developed software showed a higher cost per line than Air Force developed. Several factors could cause this. First, contractors are driven by profits. They must charge more than their costs. This would usually mean that contractor developed software would cost more than Air Force developed. It could also be caused by better tracking of costs on the part of the contractors than just more efficient programming on the part of the Air Force. Also, the fact that the contractor developed software had a lower percentage of maintenance cost to development cost than the Air Force may indicate more effective software development or better maintainability built into this software.

"Off-the-shelf" or vendor supplied software cost benefits could not be shown due to an inadequate sample size.

At the present rate, based upon this study and the literature search, an increase in the software budget and the number of programmers will be required to maintain the present development efforts. This is due to the increasing software maintenance requirements placed on organizational resources.

Recommendations

Several recommendations are being made as a result of this thesis. First, the purpose and intentions of

AFR 700-19 should be evaluated. The current purpose is to serve as a software clearinghouse. Nevertheless, this database is the only database in the DoD and AF that contains software ADPE cost data. This thesis has shown that the data contained in the ISD database does not accurately reflect true software development and maintenance costs.

This database could be of immeasurable value to the AF if three following actions were implemented:

1. Modify the database to contain fixed length fields with well-defined values for each field. This thesis analyzed only a small subset of the fields available in this database. AFR 700-19 contains many fields, besides the ones used in this thesis, that describe the hardware, software, and users of a computer system. This information can provide essential but currently unavailable cost analysis in many areas of AF computer software. Well defined data fields would help ensure validity of the data.

2. AFR 700-19 should be modified to reflect more rigorous procedures for collecting and reporting software development and maintenance costs. This would ensure that the data submitted by the major commands is more correct, complete, and consistent.

3. SISC should develop a set of statistical analysis to track software cost trends. This information could then be used by the major commands to more effectively allocate software resources.

Second, it is recommended that commands define the responsibility for implementation of the AFR 700-19 methodology to collect and document their cost data. Referring back to Henderson and Sullivan, "You cannot control what you do not measure [emphasis added]" (18:1). The costs derived in this thesis indicate that the commands do not have this information readily available and indeed they may have used the formula provided in AFR 700-19 instead. These methodologies must be concise because programmers have better things to do than fill out accounting forms. Automated accounting or "tracking" programs are becoming common today for a wide variety of hardware systems. These programs could be used to acquire the specific information needed for AFR 700-19.

Third, it is recommended that in the interim, until valid cost data can be acquired, commands with rising maintenance costs take advantage of the information in Chapter II to reduce costs. Appendices G, H, and I provide a more in-depth discussion of these concepts.

Last, it is recommended that continued research be performed on Air Force software maintenance to answer two questions:

1. What benefits occur through improvement of maintainability of existing software?

2. What benefits are received through emphasizing maintainability of AF software during software development?

Appendix A: Definition of Terms

Software Maintenance--the performance of those activities required to keep a software system operational and responsive to its users after it is accepted and placed into production. Maintenance does not include coverting software from one machine to another or from one programming language to another (15:7). Software Maintenance can be broken down into three areas:

a. Perfective--all changes, insertions, deletions, modifications, extensions, and enhancements made to meet evolving and or expanding needs of user. (Ex - making code easier to understand, improving documentation, optimizing the code, adding a new capability.)

b. Adaptive--all changes which are initiated as a result in changes in the environment. (Ex - new version of an operating system, new peripheral added to a computer system, new version of a compiler.)

c. Corrective--all changes necessitated by errors in the system. (Ex - quick fixes and firefighting, aborts because of inability to handle inputs.)

There is a great deal of overlap between these activities, and the skills required of programmers are similar.

Appendix B: SAS Program

```
COPYRIGHT (C) 1985 SAS INSTITUTE INC., CARY, N. C.  27511,
  U.S.A.
NOTE: VMS VERSION OF SAS RELEASE 5.03 AT AIR FORCE INSTI-
  TUTE OF TECHNOLOGY (03855004).
NOTE: LICENSED CPUID MODEL = 11/780, SERIAL = 01384A2A.

DATA THEFILE;
INPUT ISD $ 1-5 YEAR 6-7 CONTRACT 9 COMMAND $ 11-22
  LANG1 $ 23-34
PROG 35-40 LINES 41-50 DEVCOST 51-63 MANCOST 64-76;
IF CONTRACT=1;
IF COMMAND='      ' THEN COMMAND='OTHER';
IF COMMAND='AAC' THEN COMMAND='OTHER';
IF COMMAND='AFAA' THEN COMMAND='OTHER';
IF COMMAND='AFAF' THEN COMMAND='OTHER';
IF COMMAND='AFCAC' THEN COMMAND='OTHER';
IF COMMAND='AFCCMS' THEN COMMAND='OTHER';
IF COMMAND='AFCCS' THEN COMMAND='OTHER';
IF COMMAND='AFDSDO' THEN COMMAND='OTHER';
IF COMMAND='AFESC' THEN COMMAND='ESC';
IF COMMAND='AFIS' THEN COMMAND='OTHER';
IF COMMAND='AFISC' THEN COMMAND='OTHER';
IF COMMAND='AFLCM' THEN COMMAND='AFLC';
IF COMMAND='AFMEA' THEN COMMAND='OTHER';
IF COMMAND='AFMFC' THEN COMMAND='OTHER';
IF COMMAND='AFMSMET' THEN COMMAND='OTHER';
IF COMMAND='AFMSC' THEN COMMAND='OTHER';
IF COMMAND='AFOTEC' THEN COMMAND='OTHER';
IF COMMAND='AFRES' THEN COMMAND='OTHER';
IF COMMAND='AFOTEC' THEN COMMAND='OTHER';
IF COMMAND='AFRES' THEN COMMAND='OTHER';
IF COMMAND='AFWL' THEN COMMAND='OTHER';
IF COMMAND='ANG' THEN COMMAND='OTHER';
IF COMMAND='ANGSC' THEN COMMAND='OTHER';
IF COMMAND='DCA' THEN COMMAND='OTHER';
IF COMMAND='DLA' THEN COMMAND='OTHER';
IF COMMAND='DNA' THEN COMMAND='OTHER'
IF COMMAND='JCS' THEN COMMAND='OTHER
IF COMMAND='JDA' THEN COMMAND='OTHER
IF COMMAND='JDSSC' THEN COMMAND='OTHER ;
IF COMMAND='AFIT' THEN COMMAND='AU';
IF COMMAND='DFSEC' THEN COMMAND='OTHER';
IF COMMAND='DSDO' THEN COMMAND='DSDO';
IF COMMAND='MMSSA' THEN COMMAND='OTHER';
IF COMMAND='MED SYS DIV' THEN COMMAND='OTHER';
```

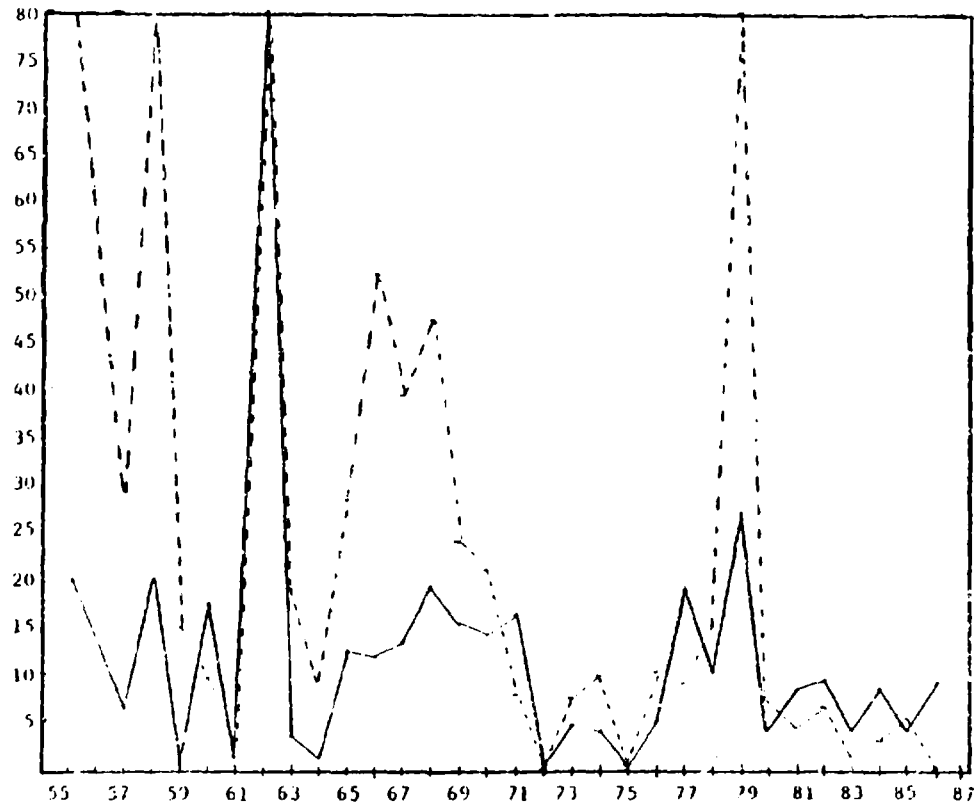
```
   IF COMMAND='SPACECMD' THEN COMMAND='OTHER';
   IF COMMAND='TACC' THEN COMMAND='TAC';
   IF COMMAND='TPSC' THEN COMMAND='OTHER';
   IF COMMAND='USAF' THEN COMMAND='OTHER';
   IF COMMAND='USAFA' THEN COMMAND='OTHER';
   IF LANG1='AFOLDS' THEN LANG1='OTHER';
   IF LANG1='ALGOL' THEN LANG1='OTHER';
   IF LANG1='APL' THEN LANG1='OTHER';
   IF LANG1='BASIC' THEN LANG1='OTHER';
   IF LANG1='BASIS' THEN LANG1='OTHER';
   IF LANG1='DAD' THEN LANG1='OTHER';
   IF LANG1='DBMS' THEN LANG1='OTHER';
   IF LANG1='GMAP' THEN LANG1='OTHER';
   IF LANG1='JOVIAL' THEN LANG1='OTHER';
   IF LANG1='PASCAL' THEN LANG1='OTHER';
   IF LANG1='PL/1' THEN LANG1='OTHER';
   IF LANG1='RPG' THEN LANG1='OTHER';
   IF LANG1='RAMIS' THEN LANG1='OTHER';
   IF LANG1='SIMSCRIPT' THEN LANG1='OTHER';
   IF LANG1='UTILITY' THEN LANG1='OTHER';
   IF LANG1='UTILITIES' THEN LANG1='OTHER';
   IF LANG1='      ' THEN LANG1='OTHER';

   CARDS;
 ;;;;
PROC SORT; BY YEAR;
PROC PRINT;
   BY YEAR;
    VAR LINES DEVCOST MANCOST;
    SUM LINES DEVCOST MANCOST;
    FORMAT LINES DEVCOST MANCOST;
 PROC SORT; BY COMMAND YEAR;
 PROC PRINT;
    BY COMMAND YEAR;
    VAR LINES DEVCOST MANCOST;
    SUM LINES DEVCOST MANCOST;
    FORMAT LINES DEVCOST MANCOST;
 PROC SORT; BY LANG1 YEAR;
 PROC PRINT;
    BY LANG1 YEAR;
    VAR LINES DEVCOST MANCOST;
    SUM LINES DEVCOST MANCOST;
    FORMAT LINES DEVCOST MANCOST;
```

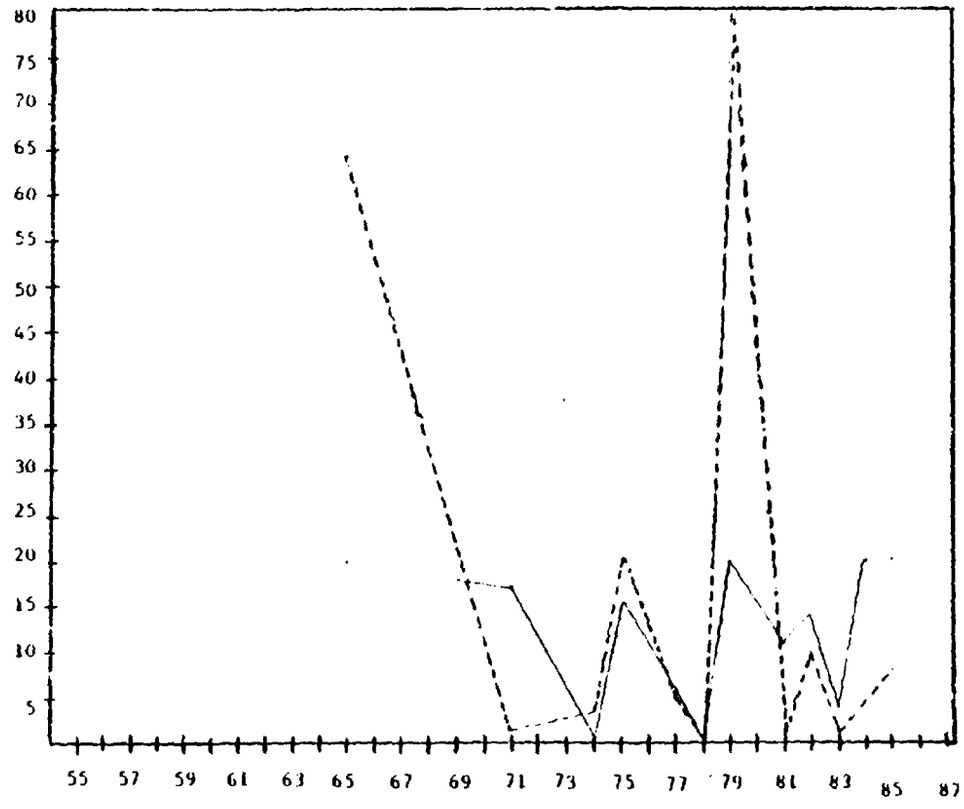
Appendix C: Cost per Line Comparison Graphs

Appendix C contains the graphs of development and maintenance cost per line over time for total AF, the four language categories, selected AF commands, and contractor developed software.

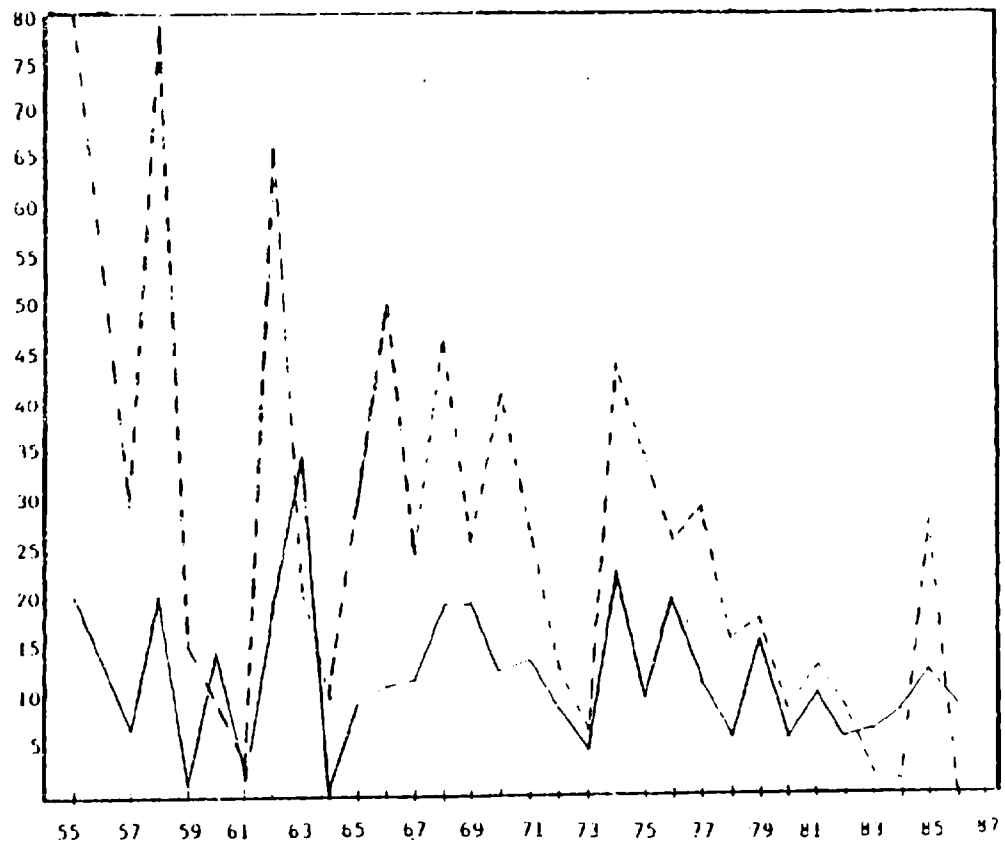
On the graph, development cost is represented by a solid line. Maintenance cost is represented by a dashed line. This legend is used on all the graphs contained in this appendix.



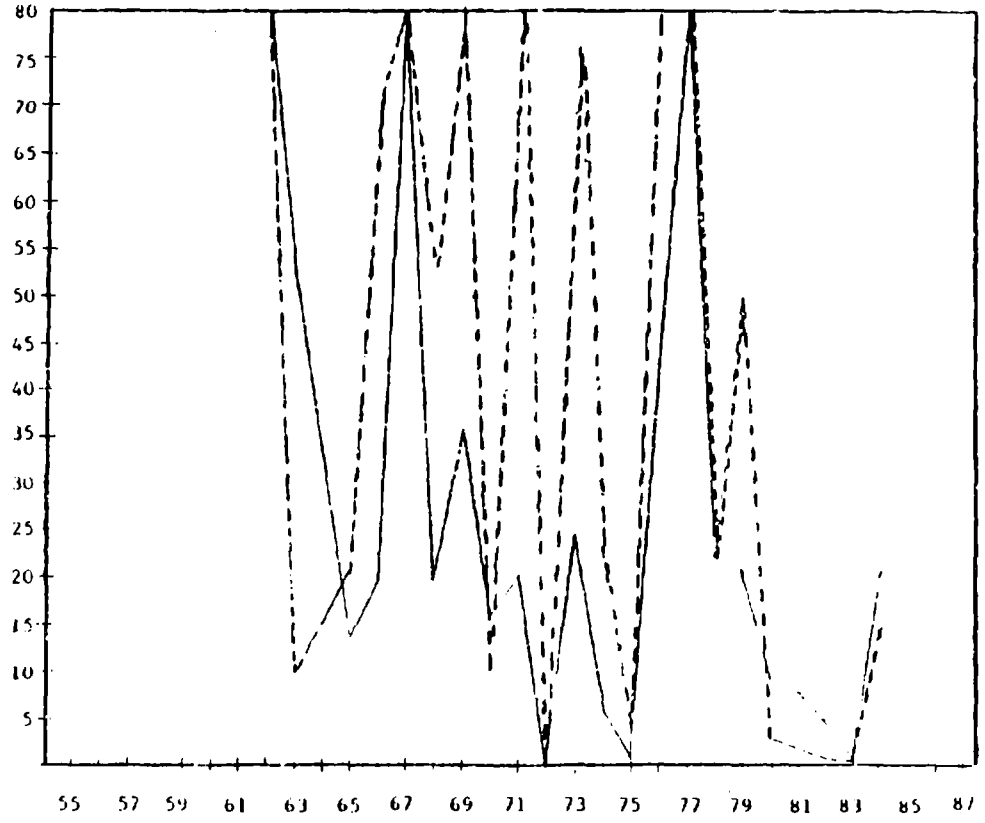
Cost per Line over Time Category - Total AF



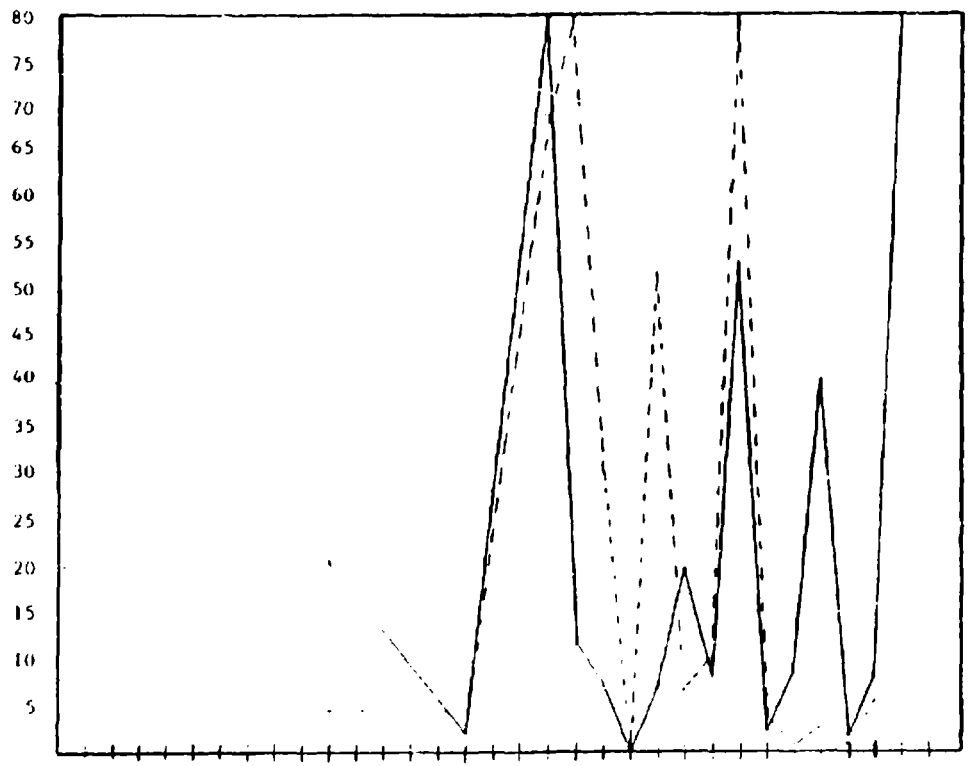
Cost per Line over Time Category - Languages - Assembler



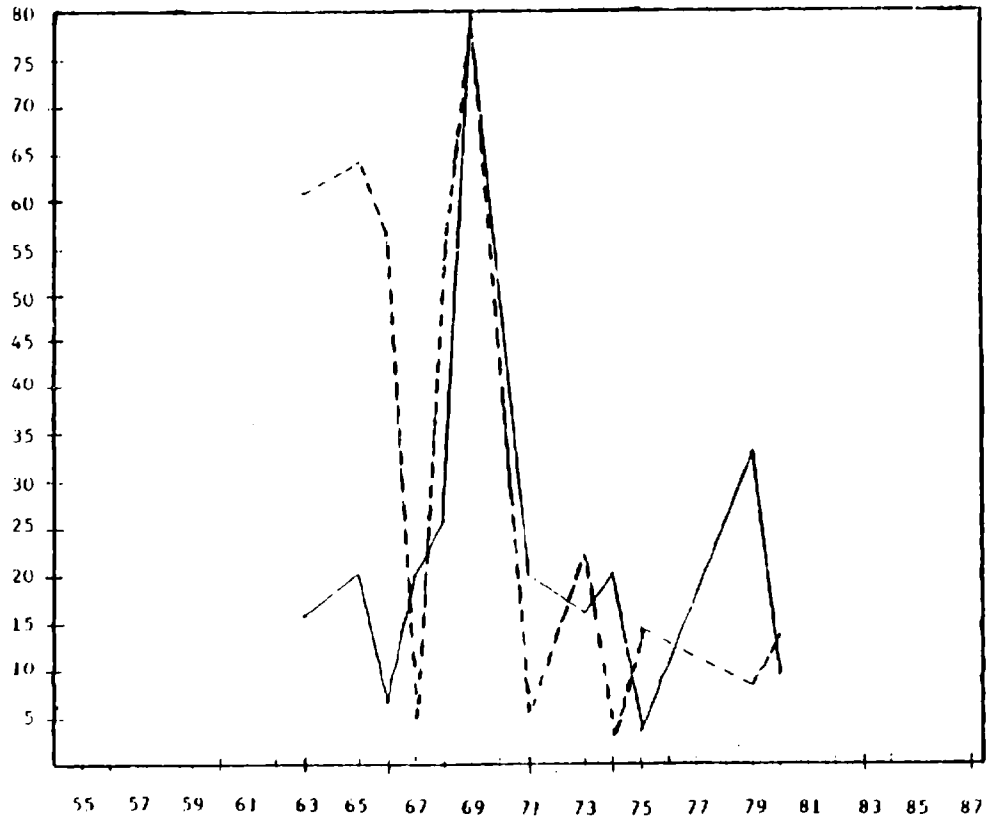
Cost per Line over Time Category - Languages - COBOL



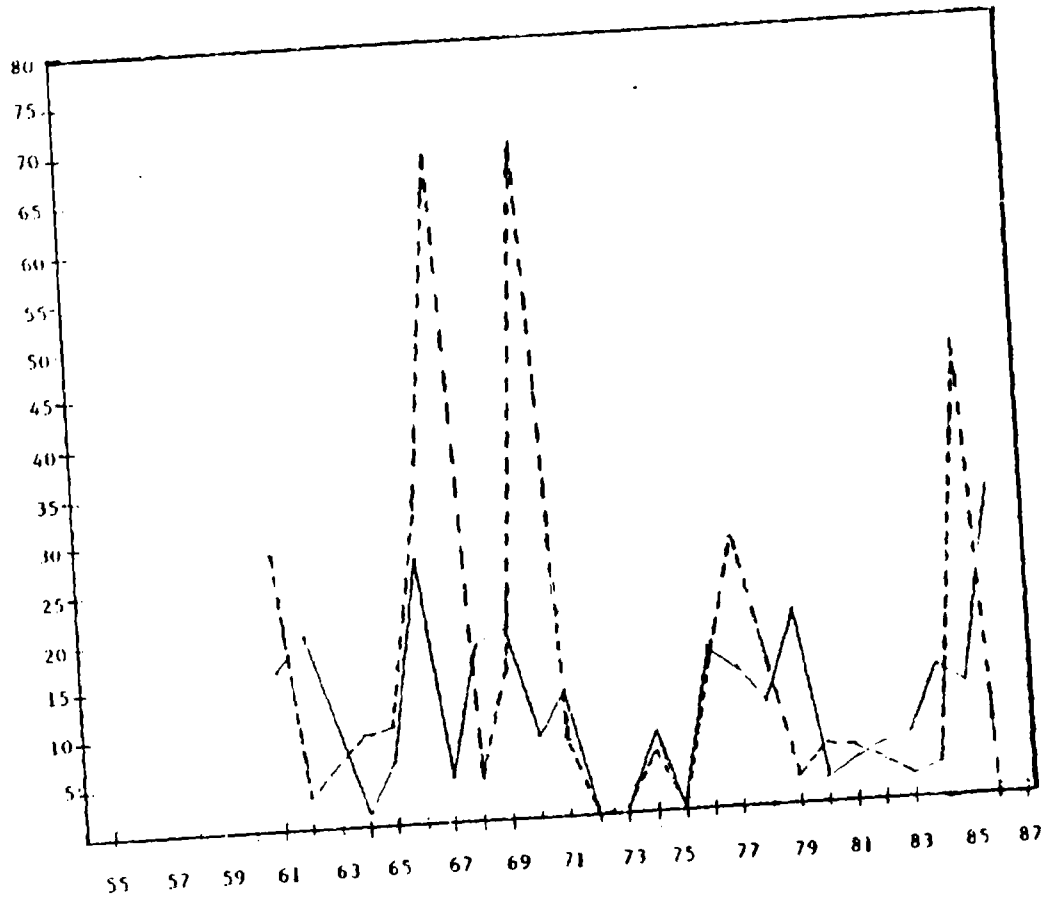
Cost per Line over Time Category - Languages - FORTRAN



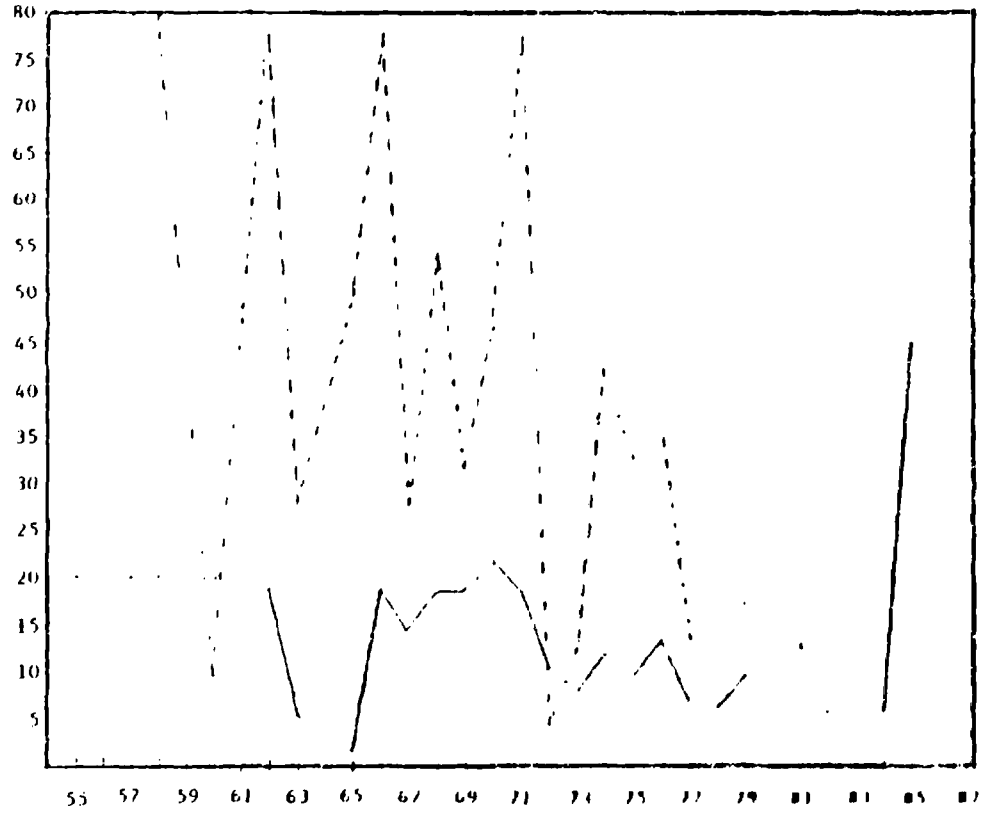
Cost per Line over Time Category - Language - Other



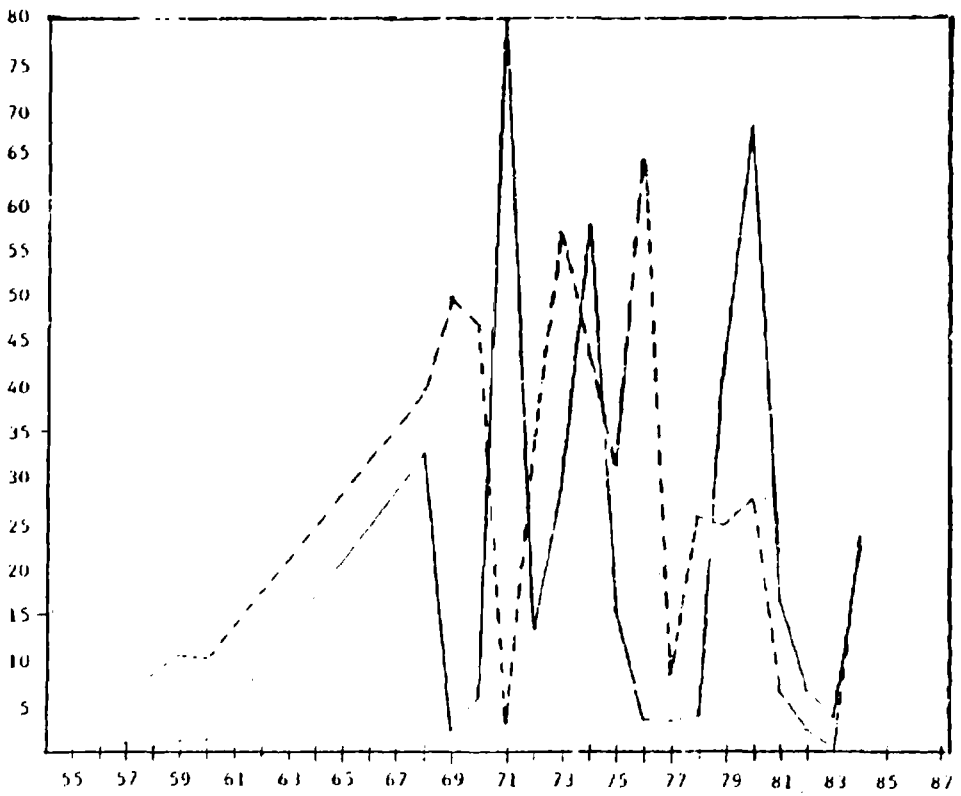
Cost per Line over Time Category - Command - AFAFC



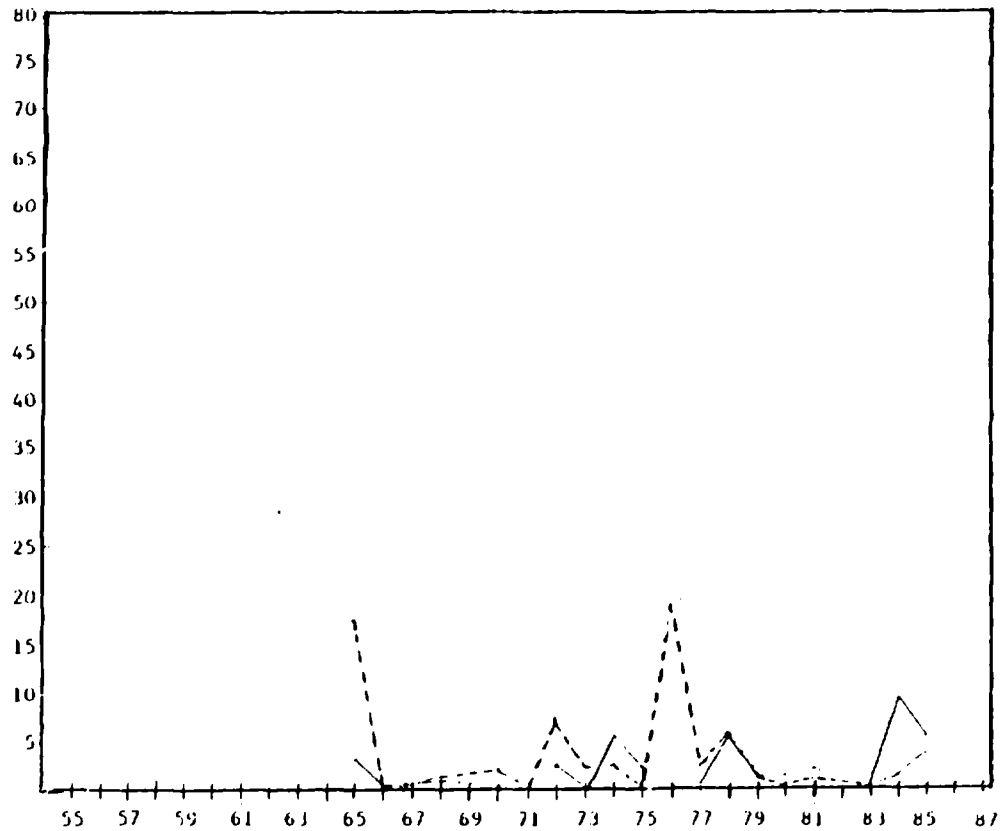
Cost per Line over Time Category - Command - AFCC



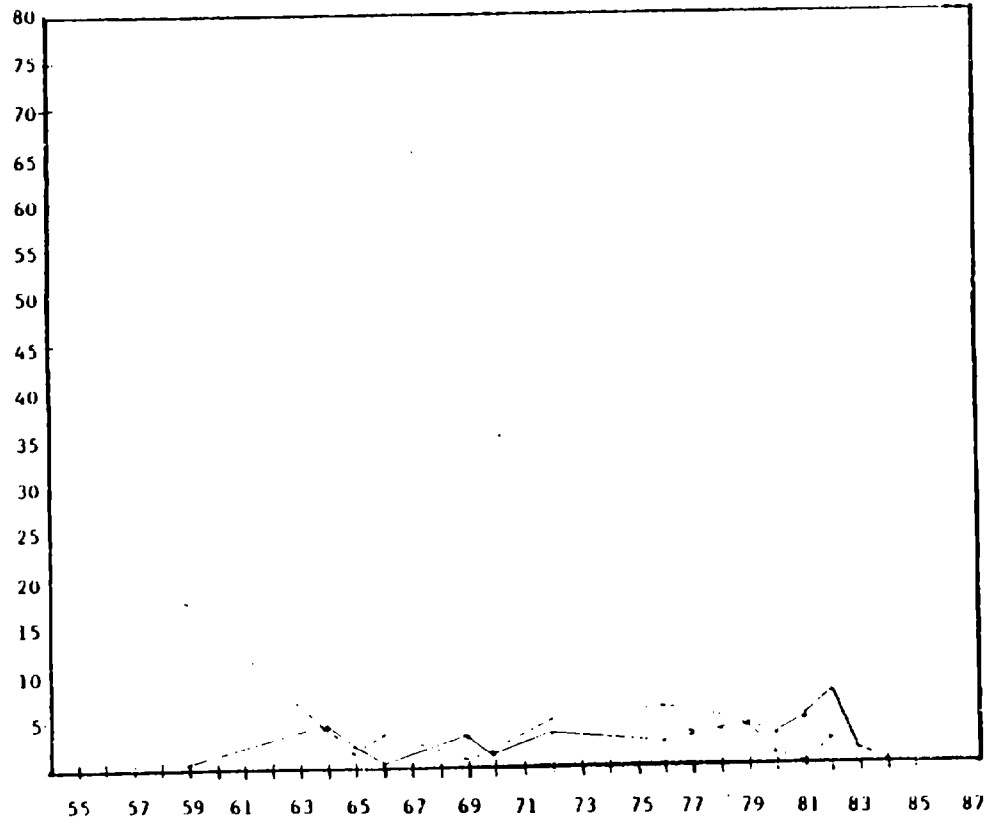
Cost per Line over Time Category - AF Commands - AFLC



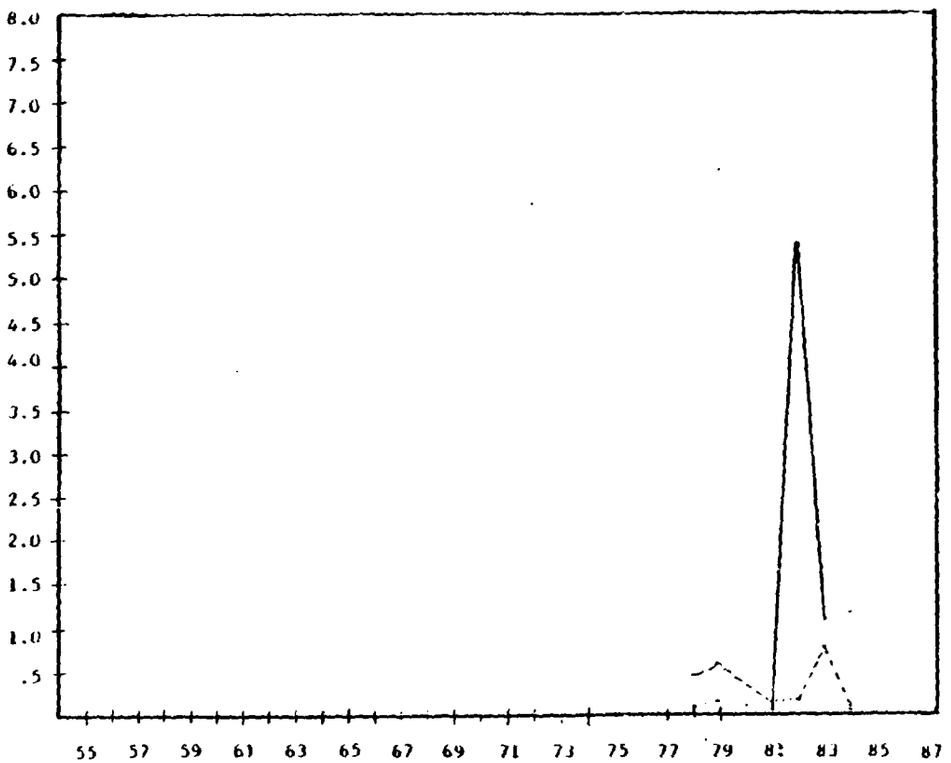
Cost per Line over Time Category - AF Commands - AFSC



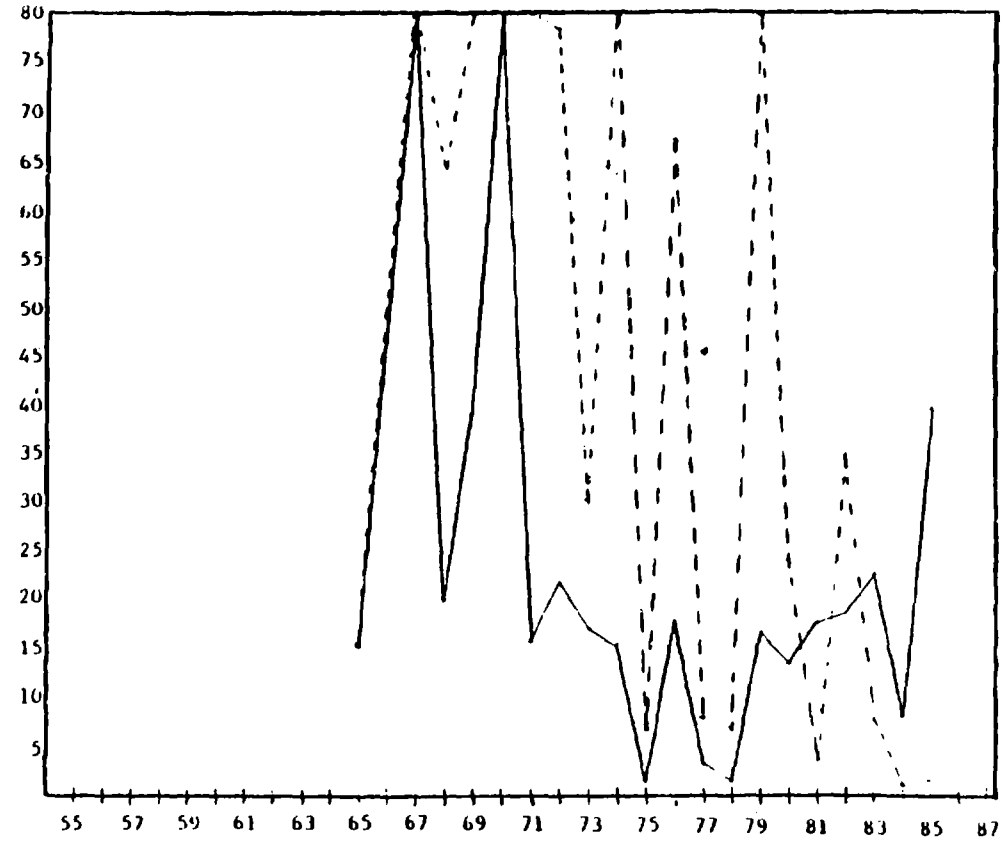
Cost per Line over Time Category - AF Commands - ATC



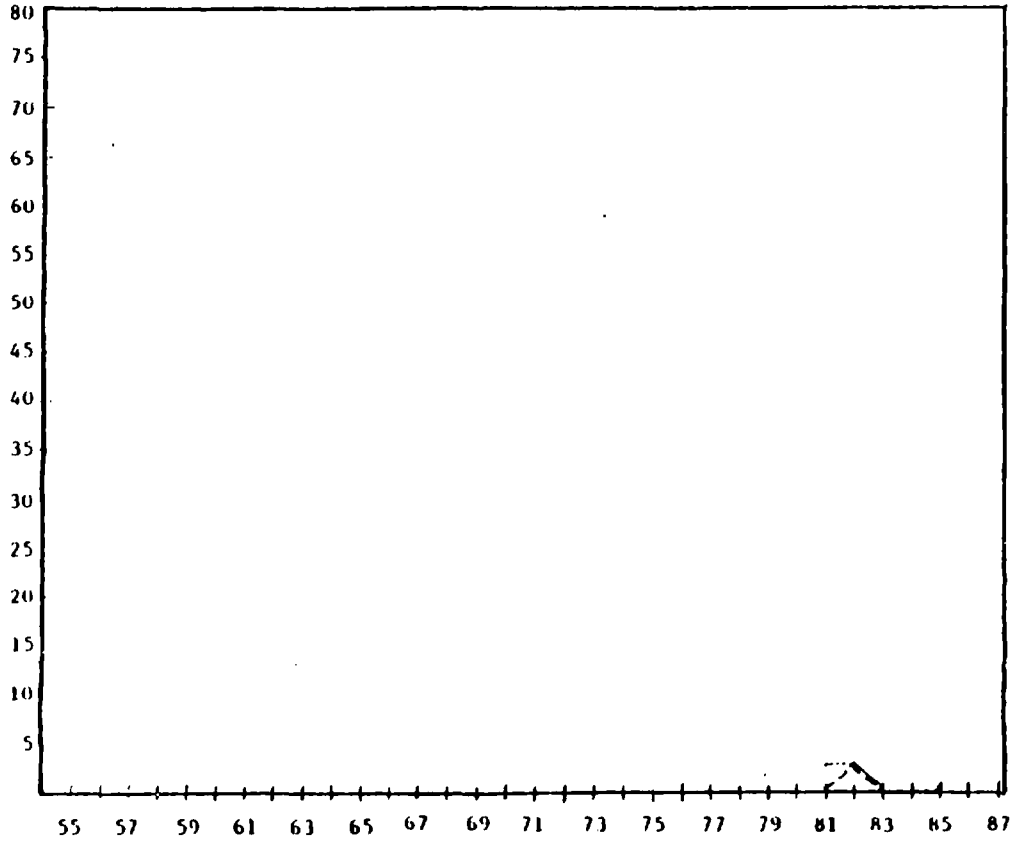
Cost per Line over Time Category - AF Commands - AU



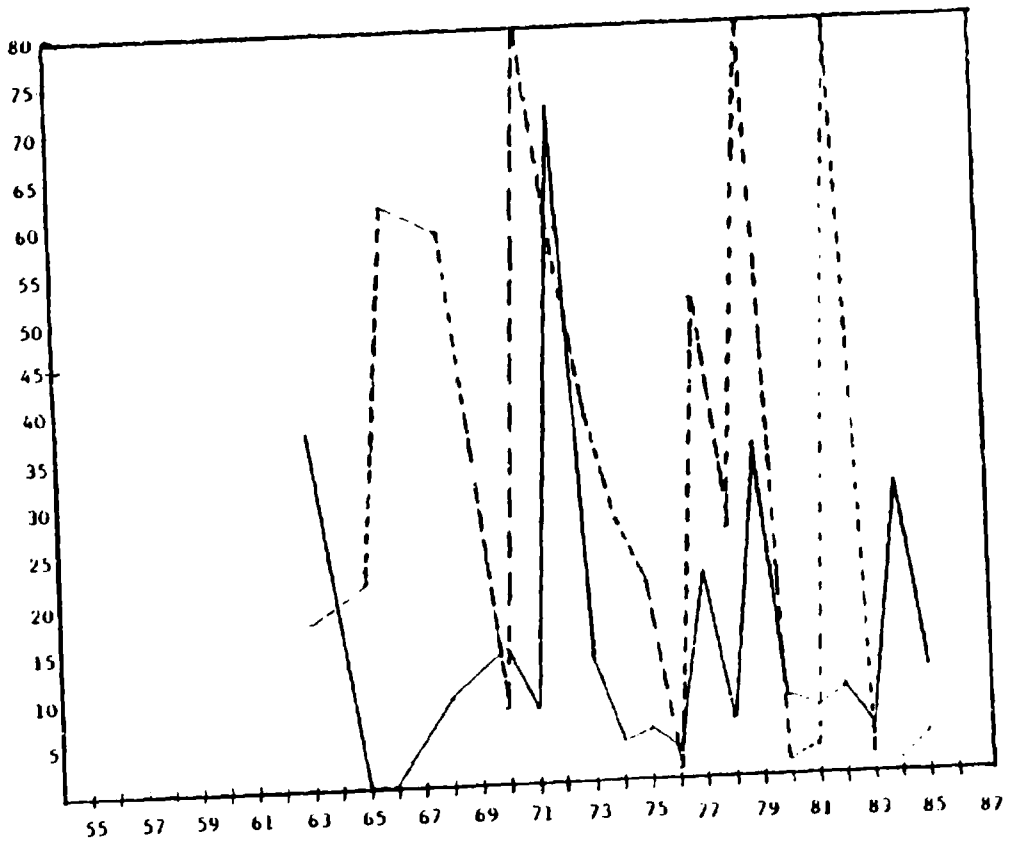
Cost per Line over Time Category - AF Commands - ESC



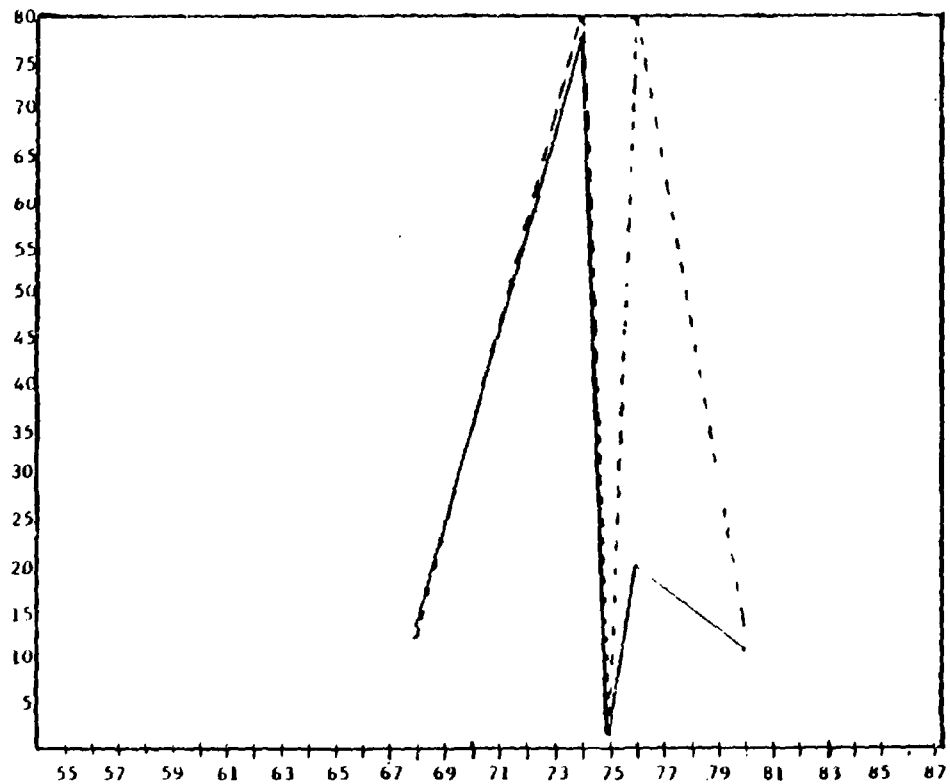
Cost per Line over Time Category - AF Commands - MAC



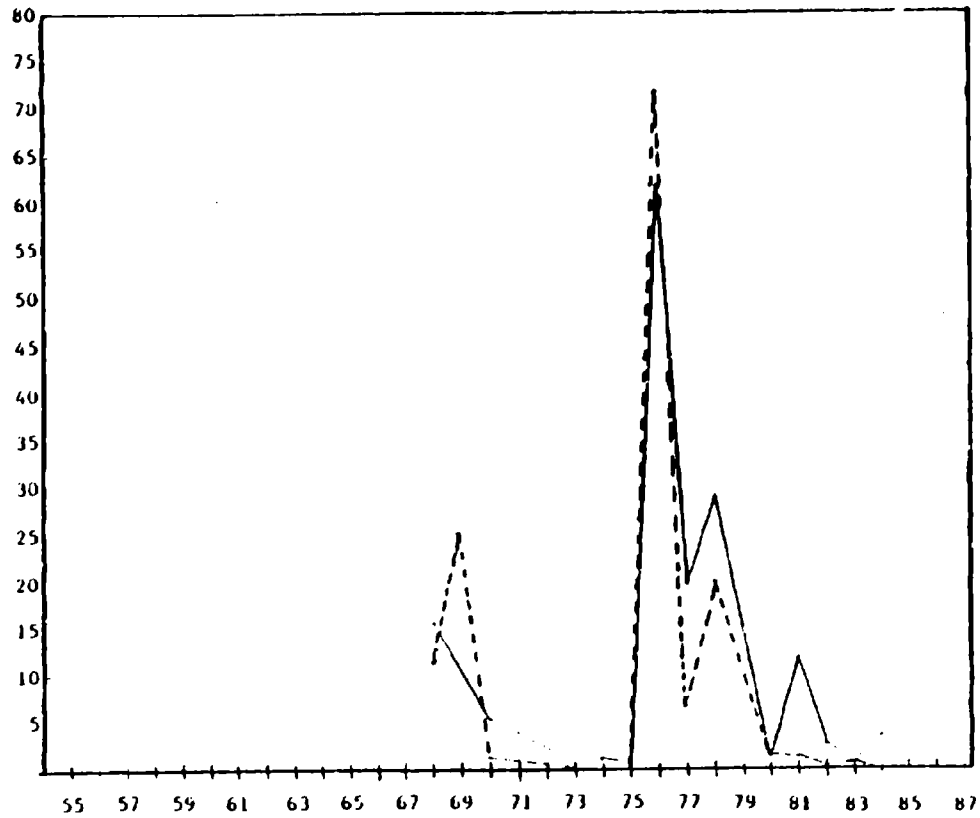
Cost per Line over Time Category - AF Commands - PACAF



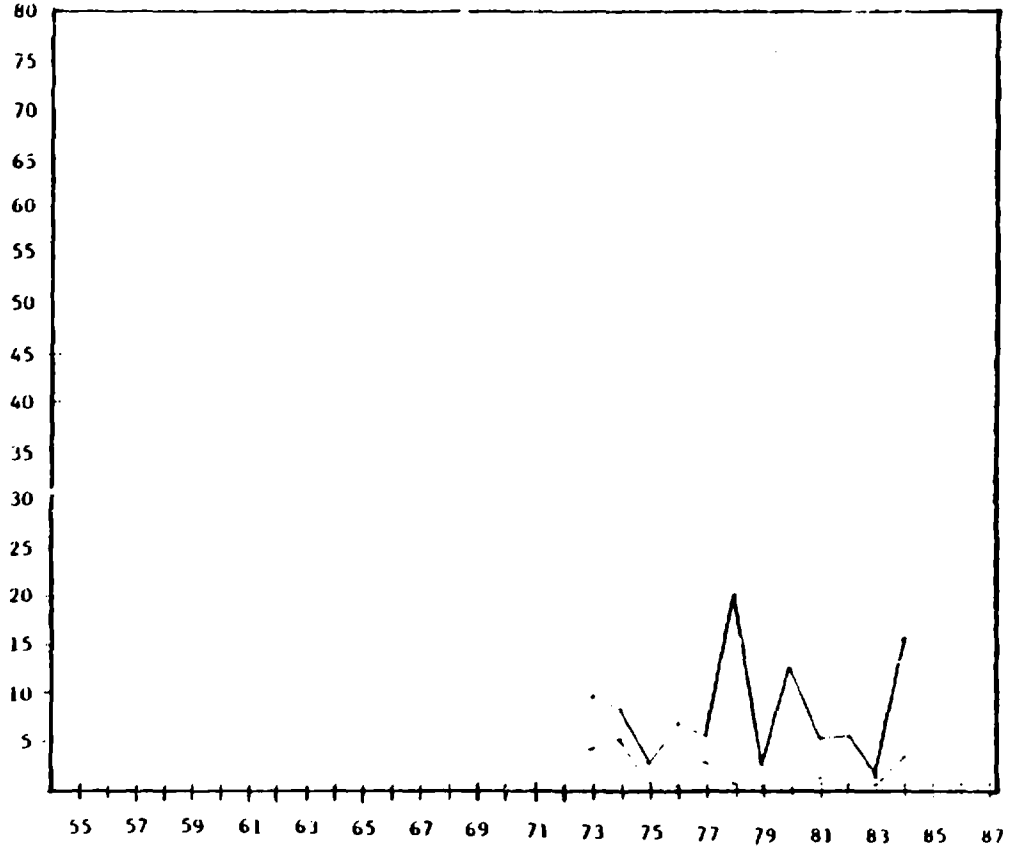
Cost per Line over Time Category - AF Commands - SAC



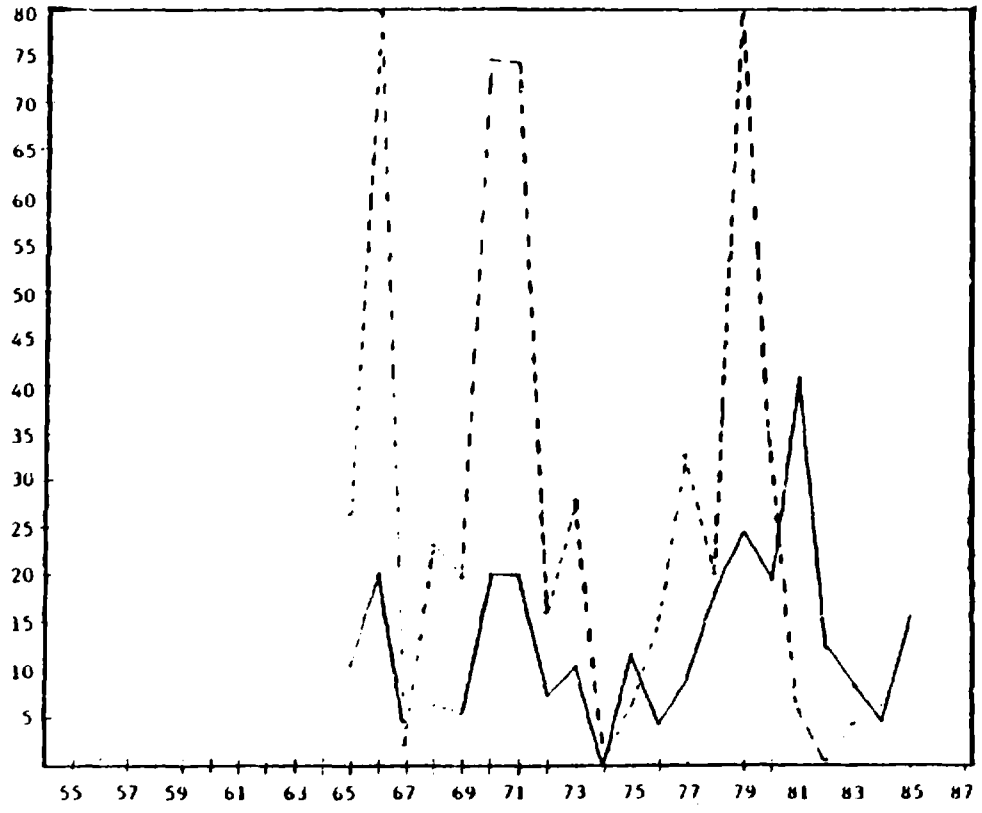
Cost per Line over Time Category - AF Commands - SISC



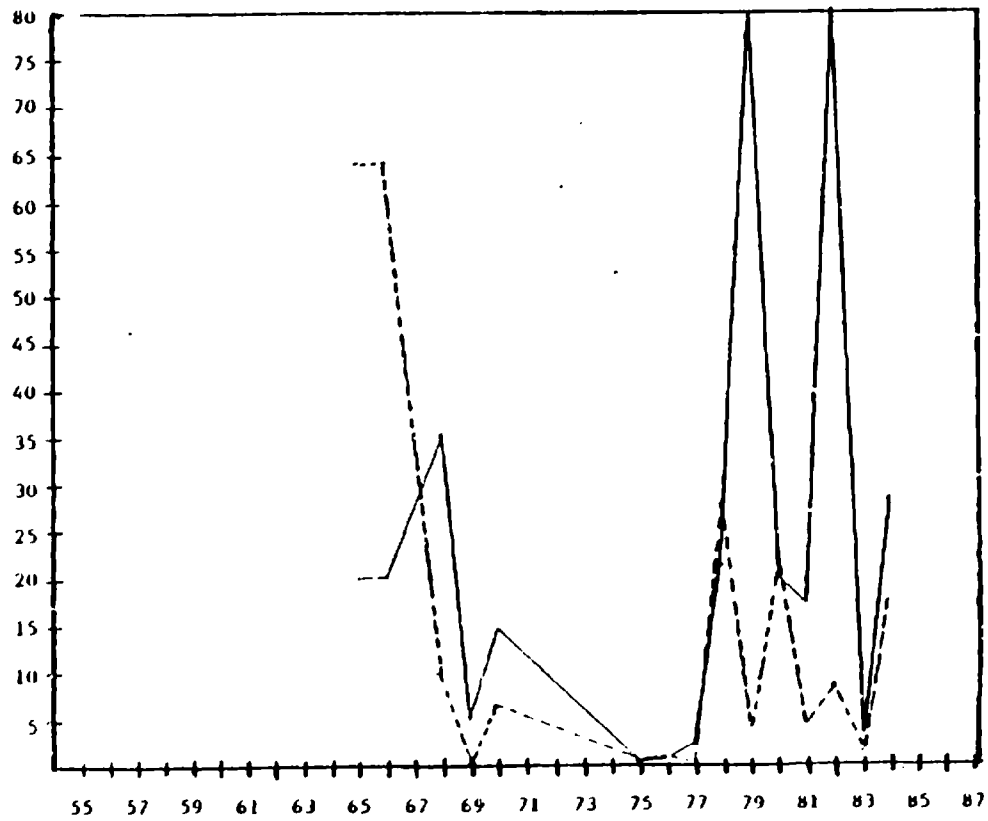
Cost per Line over Time Category - AF Commands - TA



Cost per Line over Time Category - AF Commands - USAFE



Cost per Line over Time Category - AF Commands - Others



Cost per Line over Time Category - Contractor

Appendix D: AFR 700-19 Database Description

I. Information System Designator

II. System Code

III. Title

- A. Acronym
- B. Sensitivity
- C. Criticality

IV. ADPE

V. Type System

VI. Responsible Offices

- A. ADPS
- B. ADS
- C. Development Center
- D. Air Staff Functional OPR
- E. Collateral Users

VII. Interface with other Systems

VIII. Documentation Reference

- A. Computer Operation Manual (OM)
- B. Users Manual (UM)
- C. Implementation Date of System

IX. General

- A. Programming Languages
- B. Number of Programs

C. Lines of Code

D. Processing Mode

E. Transition

X. Commercial Software Used

A. Vendor

B. Title

C. Acquisition Basis

D. Cost

XI. ADS Investment Costs

A. Initial Development Cost

B. Maintenance Cost

C. Total Cost thru FY 84

XII. Authorizing Directive

XIII. Functional Description of System

XIV. Research Results

Appendix E: Tables of Cost Totals

Appendix E contains tables developed from the data. It is organized according to category and year. The number of lines of code developed during that year, the total development cost, the development cost per line of code, the accumulated maintenance cost for the systems developed that year, and the maintenance cost per line of code. This information is organized by total AF, the four language categories, selected AF commands, and contractor developed software.

TOTAL AIR FORCE

YEAR	LINES OF CODE	DEVELOPMENT COST	DEVELOPMENT COST PER LINE	MAINTENANCE COST	MAINTENANCE COST PER LINE
55	133,075	$ 2,661,500	$ 20.00	$ 10,926,057	$ 82.10
57	77,827	505,190	6.49	2,272,159	29.19
58	91,649	1,832,980	20.00	7,331,920	80.00
59	80,335	76,000	.95	1,204,000	14.99
60	220,147	3,845,000	17.47	2,113,393	9.60
61	1,260,150	1,950,000	1.55	3,366,107	2.67
62	127,161	29,705,220	233.60	24,754,306	194.67
63	3,684,488	135,710,067	3.68	69,911,116	18.97
64	311,156	478,356	1.54	2,943,616	9.46
65	931,394	11,576,052	12.43	26,358,455	28.30
66	797,538	9,595,315	12.03	41,633,253	52.20
67	424,596	5,753,866	13.55	16,794,565	39.55
68	418,463	8,037,412	19.21	19,804,720	47.37
69	1,676,859	26,620,008	15.87	40,212,804	23.98
70	1,203,452	17,483,815	14.53	25,268,538	21.00
71	1,692,047	27,427,601	16.21	13,229,110	7.82
72	502,056,301	132,804,183	.26	32,880,607	.07
73	12,444,434	61,940,602	4.98	98,216,492	7.89
74	5,074,733	22,018,883	4.34	51,079,980	10.07
75	51,902,579	13,811,188	.27	35,067,770	.68
76	9,592,159	53,633,653	5.59	97,937,392	10.21
77	11,382,547	216,680,497	19.04	106,956,250	9.40
78	1,644,848	17,197,293	10.46	25,193,840	15.32
79	3,245,477	87,437,011	26.94	478,255,578	147.36
80	4,643,429	28,992,290	4.52	33,879,384	7.30
81	9,456,575	84,115,921	8.89	4,342,842	4.59
82	4,862,947	46,746,699	9.61	33,307,362	6.85
83	2,807,101	12,141,764	4.33	4,058,407	1.45
84	4,525,394	39,062,867	8.63	13,779,709	3.04
85	15,350,979	63,037,789	4.11	83,642,709	5.45
86	176,595	1,605,720	9.09	9,726	.06
TOTAL	639,752,176	$1,135,073,357	$ 1.77	$1,403,966,298	$ 2.19
GRAND TOTAL	653,760,528	1,211,161,856	1.85	1,489,294,836	2.27

Tables of Cost Totals - Total AF

ASSEMBLER

YEAR	LINES OF CODE	DEVELOPMENT COST	DEVELOPMENT COST PER LINE	MAINTENANCE COST	MAINTENANCE COST PER LINE
65	77,655	$ 1,553,100	$ 20.00	$ 4,969,920	$64.00
71	1,288,889	22,079,198	17.13	2,218,421	1.72
74	4,240,075	4,006,804	.95	15,285,617	3.61
75	205,127	3,171,856	15.46	4,168,174	20.32
76	7,025,000	---	---	---	---
77	500	3,000	6.00	2,500	5.00
78	56,080	27,500	.49	21,267	.38
79	1,787	35,740	20.00	142,960	80.00
81	322,914	3,558,280	11.02	360,366	1.11
82	5,010	700	.14	500	.10
83	30,000	119,342	3.98	39,070	1.30
84	8,489	169,780	20.00	---	---
85	60,000	1,200,000	20.00	480,000	8.00
TOTAL	13,321,326	$ 35,925,300	$ 2.70	$27,689,295	$ 2.08

Tables of Cost Totals - Languages - Assembler

COBOL

YEAR	LINES OF CODE	DEVELOPMENT COST	DEVELOPMENT COST PER LINE	MAINTENANCE COST	MAINTENANCE COST PER LINE
55	133,075	$ 2,661,500	$ 20.00	$10,926,057	$82.10
57	77,877	505,190	6.49	2,272,159	29.19
58	91,649	1,832,980	20.00	7,331,920	80.00
59	80,335	76,000	.95	1,204,000	14.99
60	220,147	3,845,000	14.47	2,113,392	9.60
61	1,260,150	1,950,000	1.55	3,366,107	2.67
62	71,061	1,333,220	18.76	4,691,826	66.03
63	3,142,233	108,330,650	34.48	64,936,449	20.54
64	291,095	67,336	.23	2,372	9.78
65	379,739	3,542,952	9.33	11,758,131	30.96
66	704,038	7,782,514	11.05	35,057,730	49.79
67	374,596	4,288,066	11.45	9,076,791	24.23
68	336,935	6,468,572	19.20	15,592,139	46.28
69	1,250,691	24,090,008	19.26	32,340,554	25.86
70	431,403	5,295,819	12.27	17,549,575	40.68
71	399,658	5,278,403	13.21	10,730,689	26.85
72	1,562,305	13,882,154	8.89	19,747,106	12.64
73	12,275,178	59,338,482	4.83	84,530,862	6.89
74	786,107	17,686,279	22.50	34,500,753	43.89
75	772,275	7,673,055	9.94	26,585,282	34.43
76	1,805,994	36,017,753	19.94	46,820,781	25.93
77	1,203,547	13,592,497	11.29	35,121,453	29.18
78	798,688	4,573,874	5.73	12,253,078	15.34
79	2,212,769	34,562,228	15.62	397,183,878	17.95
80	3,667,683	19,823,518	5.40	30,867,028	8.42
81	1,517,606	15,431,031	10.17	19,872,790	13.09
82	3,481,937	20,217,498	5.81	31,102,778	8.93
83	1,699,214	10,681,418	6.29	3,398,133	2.00
84	2,774,154	23,971,138	8.64	3,212,109	1.16
85	2,003,437	24,576,355	12.27	55,953,414	27.93
86	176,595	1,605,720	9.09	9,726	.06
87	450,000	33,500,000	74.44	---	---
TOTAL	46,432,091	$ 514,481,209	$ 11.08	$1,032,550,662	$22.24

Tables of Cost Totals - Languages - COBOL

FORTRAN

YEAR	LINES OF CODE	DEVELOPMENT COST	DEVELOPMENT COST PER LINE	MAINTENANCE COST	MAINTENANCE COST PER LINE
62	56,100	$ 28,372,000	$ 505.74	$ 20,062,480	$ 357.62
63	542,255	27,379,417	50.49	5,374,667	9.91
64	---	---	---	---	---
65	474,000	6,480,000	13.67	9,630,404	20.32
66	92,500	1,812,801	19.39	6,579,523	70.37
67	50,000	1,456,000	291.20	7,701,774	1,540.36
68	81,528	1,568,840	19.24	4,212,581	51.67
69	50,500	1,780,000	35.25	6,151,000	121.80
70	772,049	12,187,996	15.79	7,718,963	10.00
71	3,500	70,000	20.00	280,000	80.00
72	500,319,396	103,111,731	.21	1,089,751	.01
73	48,900	1,181,000	24.15	3,714,800	75.97
74	17,736	104,760	5.91	358,420	20.21
75	805,177	840,377	1.04	3,313,814	4.12
76	340,100	14,620,000	42.99	29,174,285	85.78
77	121,000	3,045,000	251.65	4,665,000	385.53
78	383,380	9,263,420	24.16	8,536,436	22.27
79	49,336	1,016,880	20.61	2,438,040	49.42
80	738,279	6,667,393	9.03	2,139,752	2.90
81	1,276,309	10,012,205	7.84	2,320,008	1.82
82	818,565	4,042,964	4.94	750,899	.92
83	279,112	99,090	.36	101,324	.36
84	50,400	1,038,327	20.60	749,400	14.87
85	---	---	---	---	---
86	---	---	---	---	---
TOTAL	507,638,379	$ 236,158,217	$ .47	$127,093,409	$ .25

Tables of Cost Totals - Languages - FORTRAN

OTHER LANGUAGES

YEAR	LINES OF CODE	DEVELOPMENT COST	DEVELOPMENT COST PER LINE	MAINTENANCE COST	MAINTENANCE COST PER LINE
64	20,061	$ 411,020	$ 20.49	$ 96,244	$ 4.80
67	370,000	750,000	2.03	1,700,000	4.59
72	174,600	15,810,298	90.55	12,043,750	68.98
73	120,396	1,421,120	11.80	9,970,830	82.82
74	30,815	221,040	7.17	935,190	30.35
75	50,[illegible],000	2,125,900	.04	1,000,000	.02
76	421,065	2,995,900	7.12	21,942,326	52.11
77	10,057,500	200,040,000	19.89	67,167,300	6.68
78	406,700	3,332,499	8.19	4,383,059	10.78
79	981,585	51,822,163	52.79	78,490,700	79.96
80	237,467	501,379	2.11	872,604	3.68
81	6,339,746	55,114,405	8.69	6,203,416	.98
82	557,405	22,485,537	40.34	1,452,184	2.61
83	798,775	1,241,914	1.56	519,880	.65
84	1,692,351	13,883,622	8.23	9,818,200	5.80
85	26,016	2,536,134	97.48	---	---
TOTAL	72,354,512	$378,702,732	$ 5.23	$216,661,632	$ 2.99

Tables of Cost Totals - Language - Other

AFAFC

YEAR	LINES OF CODE	DEVELOPMENT COST	DEVELOPMENT COST PER LINE	MAINTENANCE COST	MAINTENANCE COST PER LINE
63	90,000	$ 1,430,000	$ 15.89	$ 5,453,000	$ 60.59
65	77,655	1,553,100	20.00	4,969,920	64.00
66	300,000	2,000,000	6.67	16,800,000	56.00
67	10,957	219,140	20.00	55,000	5.02
68	101,200	2,615,760	25.85	5,423,600	53.59
69	5,656	512,180	90.56	1,607,240	284.17
71	27,957	559,140	20.00	155,000	5.54
73	1,506,964	24,139,280	16.02	33,835,000	22.45
74	2,499	49,980	20.00	7,500	3.00
75	83,412	297,679	3.57	1,197,762	14.36
79	25,000	821,629	32.87	200,000	8.00
80	441,050	4,139,400	9.39	5,921,800	13.43
TOTAL	2,705,450	$ 45,218,985	$ 16.71	$80,746,622	$ 29.84

Tables of Cost Totals - Command - AFAFC

AFCC

YEAR	LINES OF CODE	DEVELOPMENT COST	DEVELOPMENT COST PER LINE	MAINTENANCE COST	MAINTENANCE COST PER LINE
61	119,224	$ 1,950,000	$ 16.36	$ 3,366,107	$ 28.23
62	10,561	211,220	20.00	38,366	3.63
64	307,529	450,580	1.47	2,911,116	9.47
65	23,633	160,000	6.77	240,000	10.16
66	83,575	2,260,000	27.04	2,658,998	31.82
67	63,833	293,876	4.60	4,369,222	68.45
68	6,569	120,000	18.27	30,000	4.57
69	762,136	14,893,760	19.54	11,879,399	15.59
70	85,281	713,073	8.36	5,944,444	69.70
71	199,511	2,600,520	13.03	1,615,837	8.10
72	500,294,176	100,938,700	.20	181,678	0.00
73	8,118,805	3,525,000	.43	4,344,289	.54
74	44,906	362,760	8.08	272,282	6.06
75	50,063,709	1,063,340	.02	1,083,339	.02
76	160,077	2,643,140	16.51	2,674,726	16.71
77	46,645	671,700	14.40	1,290,282	27.66
78	131,137	1,405,489	10.72	2,059,288	15.70
79	645	12,900	20.00	1,900	2.95
80	2,727,903	6,048,820	2.60	13,990,563	6.01
81	81,606	390,228	4.78	487,042	5.97
82	1,142,503	6,940,720	6.08	4,718,193	4.13
83	585,725	4,026,724	6.87	1,394,118	2.38
84	437,757	6,031,520	13.78	1,664,000	3.80
85	1,121,200	13,456,550	11.90	52,800,000	46.68
86	176,595	5,605,720	31.74	9,726	.06
TOTAL	544,854,741	$176,778,340	$ .31	$120,024,885	$ .21

Tables of Cost Totals - Command - AFCC

AFLC

YEAR	LINES OF CODE	DEVELOPMENT COST	DEVELOPMENT COST PER LINE	MAINTENANCE COST	MAINTENANCE COST PER LINE
55	133,075	$ 2,661,500	$ 20.00	$ 10,926,057	$ 82.10
57	22,222	444,440	20.00	1,921,159	86.45
58	91,649	1,832,980	20.00	7,331,920	80.00
60	190,000	3,800,000	20.00	1,802,893	9.49
62	66,666	1,244,000	18.68	5,141,490	77.20
63	50,501	309,407	6.13	1,381,446	27.35
65	200,446	2,871,140	1.40	10,031,506	50.05
66	239,566	4,192,801	18.73	18,684,465	77.99
67	313,591	4,617,730	14.73	8,594,754	27.41
68	64,188	1,171,240	18.25	3,467,176	54.02
69	754,106	6,545,260	18.48	11,086,375	31.31
70	227,207	4,901,936	21.57	10,633,518	46.80
71	99,595	1,030,000	18.53	4,292,018	77.20
72	886,100	9,332,509	10.53	3,708,293	4.18
73	1,399,463	11,152,012	7.97	17,554,353	12.54
74	400,[illegible]7	4,867,718	12.16	16,745,216	41.84
75	725,969	6,746,212	9.29	23,427,475	32.27
76	684,813	9,061,066	13.23	24,014,149	35.07
77	431,124	3,061,133	7.10	5,783,788	13.42
78	654,083	3,953,994	6.05	12,136,764	18.54
79	688,649	6,521,457	9.47	11,918,883	17.31
80	259,800	1,854,187	7.14	3,281,125	12.63
81	1,205,638	6,274,458	5.20	15,922,882	13.21
82	1,689,337	9,678,832	5.73	7,535,248	4.46
83	724,356	3,238,010	4.47	839,112	1.15
84	1,369,405	7,748,540	5.66	1,007,072	.74
85	78,661	3,570,925	45.40	---	---
TOTAL	13,473,628	$156,491,502	$ 11.61	$239,180,293	$ 17.75

Tables of Cost Totals - AF Commands - AFLC

AFSC

YEAR	LINES OF CODE	DEVELOPMENT COST	DEVELOPMENT COST PER LINE	MAINTENANCE COST	MAINTENANCE COST PER LINE
57	55,615	$ 60,750	$ 1.09	$ 351,000	$ 6.31
59	30,219	36,000	1.19	324,000	10.72
60	30,147	45,000	1.49	310,500	10.30
68	22,800	742,000	32.54	905,000	39.69
69	10,500	30,000	2.86	521,000	49.62
70	19,358	119,000	6.15	903,500	46.67
71	1,064,882	22,484,280	211.18	3,508,921	3.30
72	97,273	1,322,132	13.59	3,332,096	34.26
73	109,804	3,126,330	28.47	6,179,180	56.27
74	186,762	10,689,010	57.23	8,040,282	43.05
75	109,162	1,699,383	15.57	3,424,876	31.37
76	350,696	1,362,374	3.88	22,626,149	64.52
77	20,400	73,000	3.58	174,300	8.54
78	54,580	270,500	4.96	1,398,723	25.63
79	15,454	643,900	41.67	383,666	24.83
80	52,947	3,609,428	68.17	1,449,179	27.37
81	43,400	708,114	16.31	299,570	6.90
82	95,000	630,206	6.63	225,775	2.38
83	123,200	480,244	3.90	51,584	.42
84	30,719	680,469	22.15	720,768	23.46
TOTAL	2,526,268	$ 48,812,120	$ 19.32	$55,130,139	$ 21.82

Tables of Cost Totals - AF Commands - AFSC

ATC

YEAR	LINES OF CODE	DEVELOPMENT COST	DEVELOPMENT COST PER LINE	MAINTENANCE COST	MAINTENANCE COST PER LINE
65	51,000	$ 167,436	$ 3.28	$ 895,405	$ 17.56
66	105,500	22,755	.22	78,500	.74
67	4,000	2,520	.63	3,636	.91
68	30,000	31,072	1.04	35,834	1.19
70	67,000	22,066	.34	147,022	2.19
71	15,000	2,823	.19	11,035	.74
72	31,726	90,680	2.86	235,907	7.44
73	86,200	35,421	.41	241,193	2.80
74	60,900	353,183	5.80	177,707	2.92
75	9,000	21,991	2.44	3,149	.35
76	315,400	131,515	.42	6,022,534	19.09
77	79,360	56,601	.71	191,527	2.41
78	21,855	126,166	5.77	127,064	5.81
79	3,722	4,361	1.17	4,934	1.33
80	552,378	883,408	1.60	401,457	.73
81	52,217	105,415	2.02	80,621	1.54
83	588,578	592,963	1.01	65,229	.11
84	88,006	829,571	9.43	145,192	1.65
85	100,000	530,631	5.31	395,000	3.95
TOTAL	2,261,862	$ 4,012,581	$ 1.77	$ 9,262,947	$ 4.10

Tables of Cost Totals - AF Commands - ATC

AU

YEAR	LINES OF CODE	DEVELOPMENT COST	DEVELOPMENT COST PER LINE	MAINTENANCE COST	MAINTENANCE COST PER LINE
59	50,116	$ 40,000	$ .80	$ 880,000	$ 17.56
64	3,627	17,976	4.96	16,500	4.55
65	99,331	238,196	2.40	166,500	1.68
66	27,000	18,000	.67	95,000	3.52
69	9,629	30,000	3.11	10,000	1.04
70	29,970	45,000	1.50	55,000	1.84
72	36,000	129,168	3.59	183,000	5.08
76	12,000	35,000	2.92	75,500	6.29
77	11,659	41,020	3.52	44,700	3.83
78	12,653	51,312	4.06	67,900	5.37
79	35,000	160,425	4.58	150,000	4.29
80	12,000	42,000	3.52	14,000	1.17
81	3,048	15,240	5.00	1,000	.33
82	3,077	24,195	7.86	8,500	2.96
83	73,749	130,090	1.76	130,219	1.77
84	17,000	1,732	.10	14,772	.87
TOTAL	452,879	$ 1,031,521	$ 2.28	$ 1,924,273	$ 4.25

Tables of Cost Totals - AF Commands - AU

ESC

YEAR	LINES OF CODE	DEVELOPMENT COST	DEVELOPMENT COST PER LINE	MAINTENANCE COST	MAINTENANCE COST PER LINE
78	250,000	$ 22,500	$ .09	$ 112,500	$ .45
79	27,000	3,900	.14	15,600	.58
81	116,344	7,700	.07	15,400	.13
82	81,696	438,850	5.37	14,250	.17
83	21,800	23,250	1.07	16,600	.76
84	513,900	590,900	1.15	16,100	.03
TOTAL	1,137,540	$ 1,207,100	$ 1.06	$ 190,450	$ .17

Tables of Cost Totals - AF Commands - ESC

MAC

YEAR	LINES OF CODE	DEVELOPMENT COST	DEVELOPMENT COST PER LINE	MAINTENANCE COST	MAINTENANCE COST PER LINE
62	---	$ 8,250,000	$ ---	$ 16,050,000	$ ---
65	396,000	6,150,000	15.53	8,060,000	20.35
67	415	464,300	1,118.80	3,701,952	8,920.39
68	138,767	2,775,340	20.00	8,881,088	64.00
69	48,069	1,889,358	39.31	5,361,540	111.54
70	557	361,140	648.37	1,173,840	2,107.43
71	11,162	179,938	16.12	1,614,582	144.65
72	78,100	1,706,130	21.85	6,126,720	78.45
73	1,096,419	18,709,089	17.06	32,876,782	29.99
74	238,520	3,702,941	15.53	21,794,221	91.37
75	552,018	1,029,950	1.87	3,850,521	6.98
76	69,262	1,236,438	17.85	4,633,327	67.18
77	162,944	522,450	3.21	1,312,027	8.05
78	42,398	70,002	1.65	297,281	7.01
79	162,421	2,712,299	16.70	18,927,490	116.81
80	89,435	1,200,406	13.42	2,134,529	24.00
81	239,732	4,229,350	17.74	792,978	3.31
82	240,568	4,447,870	18.49	9,549,578	39.70
83	57,295	1,298,293	22.66	438,405	7.65
84	41,889	499,380	8.07	61,725	1.00
85	65,221	2,590,275	39.72	80,000	1.23
TOTAL	3,751,192	$64,024,749	$ 17.07	$147,795,635	$ 39.40

Tables of Cost Totals - AF Commands - MAC

PACAF

YEAR	LINES OF CODE	DEVELOPMENT COST	DEVELOPMENT COST PER LINE	MAINTENANCE COST	MAINTENANCE COST PER LINE
81	3,085,050	$8,846,000	$ 2.87	$ 750,000	$ .24
82	7,940	23,767	2.99	17,952	2.26
83	15,900	7,242	.52	1,000	.07
85	11,923	2,190	.18	223	.02
TOTAL	3,163,596	$8,879,199	$ 2.81	$ 769,175	$ .24

Tables of Cost Totals - AF Commands - PACAF

SAC

YEAR	LINES OF CODE	DEVELOPMENT COST	DEVELOPMENT COST PER LINE	MAINTENANCE COST	MAINTENANCE COST PER LINE
63	3,543,987	$133,970,660	$ 37.80	$ 63,076,670	$ 17.80
65	45,000	30,000	.67	984,404	21.88
66	1,897	759	.40	116,290	61.30
68	2,731	27,000	9.89	160,700	58.84
70	753,000	11,180,000	14.85	6,229,640	8.27
71	851	6,900	8.11	76,350	89.95
72	225,926	16,194,865	71.68	12,590,247	55.73
73	11,415	153,980	13.48	441,075	38.64
74	21,910	101,700	4.64	610,778	27.88
75	12,855	68,820	5.35	269,965	20.98
76	7,564,039	24,138,000	3.19	8,477,650	1.12
77	484,000	10,400,000	21.49	24,500,000	50.62
78	4,543	28,800	6.34	120,530	26.53
79	244,284	8,560,800	35.04	304,135,000	1,245.02
80	663,641	5,658,040	8.53	1,182,740	1.78
81	1,426,265	10,276,485	7.21	4,755,900	3.33
82	12,718	115,800	9.11	9,113,308	716.57
83	341,304	1,722,947	5.05	736,675	2.16
84	492,591	15,248,680	30.96	805,580	1.64
85	159,087	1,786,963	11.23	746,509	4.69
TOTAL	16,062,044	$259,671,199	$ 16.17	$442,654,461	$ 27.56

Tables of Cost Totals - AF Commands - SAC

SISC

YEAR	LINES OF CODE	DEVELOPMENT COST	DEVELOPMENT COST PER LINE	MAINTENANCE COST	MAINTENANCE COST PER LINE
68	14,420	$ 200,000	$ 13.87	$ 176,800	$ 12.26
74	14,581	1,130,763	77.56	2,014,560	138.17
75	6,926	13,200	1.91	236,920	3.89
76	200,000	4,000,000	20.00	16,000,000	80.80
80	33,153	364,320	10.99	445,680	13.44
TOTAL	269,080	$5,708,283	$ 21.21	$18,873,960	$ 70.14

Tables of Cost Totals - AF Commands - SISC

TAC

YEAR	LINES OF CODE	DEVELOPMENT COST	DEVELOPMENT COST PER LINE	MAINTENANCE COST	MAINTENANCE COST PER LINE
68	12,652	$ 200,000	$ 15.81	$ 145,860	$ 11.53
69	5,668	---	---	141,250	24.92
70	19,049	100,000	5.25	30,000	1.57
71	288,889	---	---	---	---
73	11,774	7,270	.62	2,986	.25
74	43,343	44,783	1.03	13,693	.32
75	70,574	26,952	.38	17,957	.25
76	174,202	10,750,000	61.71	12,490,000	71.70
77	10,041,814	200,022,937	19.92	67,035,385	6.68
78	283,763	8,151,350	28.73	5,609,910	19.77
80	40,847	49,409	1.21	57,752	1.41
81	2,794,883	32,869,830	11.76	3,435,091	1.23
82	20,871	56,727	2.72	3,656	.18
83	148,568	128,300	.86	126,423	.85
84	29,396	130,072	3.30	1,900	.05
TOTAL	14,210,651	$252,537,630	$ 17.77	$ 89,111,862	$ 6.27

Tables of Cost Totals - AF Commands - TAC

USAFE

YEAR	LINES OF CODE	DEVELOPMENT COST	DEVELOPMENT COST PER LINE	MAINTENANCE COST	MAINTENANCE COST PER LINE
73	6,770	$ 66,420	$ 9.81	$ 31,234	$ 4.61
74	26,734	221,240	8.28	145,508	5.44
75	41,424	122,160	2.95	40,956	.99
76	2,320	16,400	7.07	1,230	.53
77	18,359	107,880	5.88	55,034	3.00
78	7,394	147,880	20.00	6,680	.90
79	26,921	74,540	2.77	12,165	.45
80	25,365	309,520	12.86	25,409	1.09
81	54,966	280,960	5.11	69,340	1.26
82	42,297	234,940	5.55	28,391	.67
83	86,126	142,000	1.65	69,500	.81
84	9,631	150,000	15.57	32,200	3.34
TOTAL	360,303	$1,917,940	$ 5.54	$ 518,647	$ 1.50
GRAND TOTAL	439,752,176	$1,135,073,357	$ 1.77	$1,403,966,298	$ 2.19

Tables of Cost Totals - AF Commands - USAFE

OTHER COMMANDS

YEAR	LINES OF CODE	DEVELOPMENT COST	DEVELOPMENT COST PER LINE	MAINTENANCE COST	MAINTENANCE COST PER LINE
65	38,309	$ 406,180	$ 10.60	$ 1,010,720	$ 26.38
66	40,000	800,000	20.00	3,200,000	80.00
67	31,800	156,300	4.92	70,000	2.20
68	25,136	155,000	6.17	578,662	23.02
69	481,095	2,719,450	5.65	9,606,000	19.97
70	2,030	40,600	20.00	151,584	74.66
71	28,200	564,000	20.00	1,955,167	69.33
72	407,000	3,090,000	7.59	6,522,666	16.03
73	96,800	1,026,000	10.60	2,710,400	28.00
74	4,034,351	494,805	.12	1,258,133	.31
75	228,030	2,721,500	11.93	1,515,100	6.64
76	58,386	259,720	4.45	902,080	15.45
77	80,746	746,076	9.24	2,658,410	32.92
78	162,442	2,969,300	18.28	3,267,200	20.11
79	1,791,040	43,266,700	24.16	142,459,900	79.54
80	144,410	2,842,152	19.68	4,963,090	34.37
81	353,426	14,567,644	41.22	2,146,357	6.07
82	1,493,840	18,253,293	12.22	882,550	.59
83	42,540	351,700	8.27	198,550	4.67
84	1,465,100	7,152,000	4.88	9,310,400	6.35
85	391,395	6,189,988	15.81	2,400,000	6.13
TOTAL	12,537,002	$108,782,208	$ 8.68	$197,782,949	$ 15.76

Tables of Cost Totals - AF Commands - Others

CONTRACTOR DEVELOPED SOFTWARE

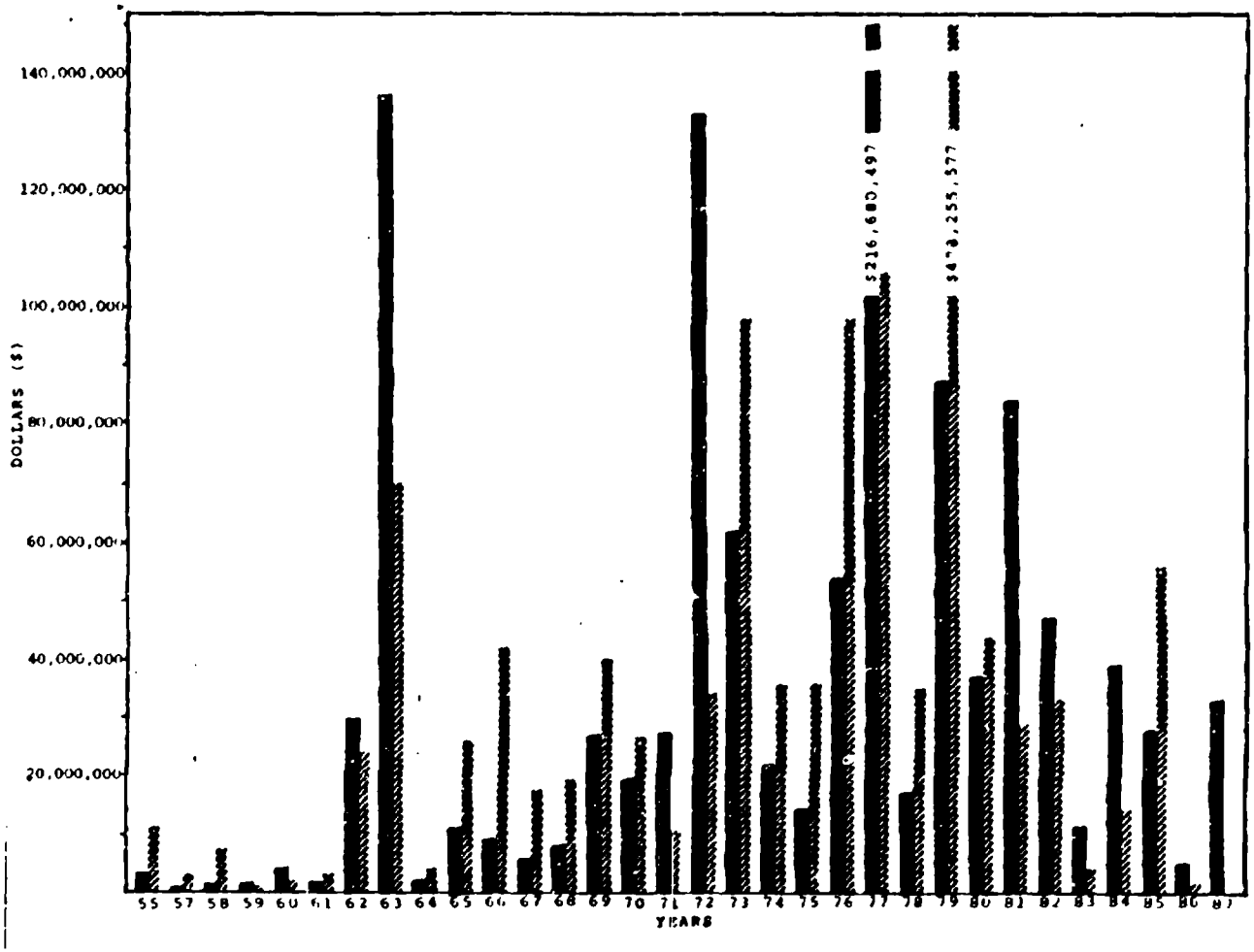
YEAR	LINES OF CODE	DEVELOPMENT COST	DEVELOPMENT COST PER LINE	MAINTENANCE COST	MAINTENANCE COST PER LINE
65	77,455	$ 1,553,100	$ 20.00	$ 4,969,920	$ 64.00
66	700,000	2,000,000	20.00	16,800,000	56.00
68	20,000	700,000	35.00	200,000	10.00
69	15,729	80,000	5.09	5,500	.35
70	750,000	11,000,000	14.67	5,000,000	6.67
75	289,412	55,636	.19	145,691	.50
76	47,700	46,000	.96	47,200	.99
77	57,500	40,000	.70	167,300	2.91
78	160,000	3,375,999	21.10	4,425,309	27.66
79	35,000	7,706,325	220.18	150,000	4.29
80	15,050	301,000	20.00	337,120	22.40
81	1,071,000	18,445,426	17.21	4,963,138	4.63
82	194,367	17,472,520	89.89	1,720,253	8.85
83	66,876	238,618	3.57	99,142	1.48
84	527,778	14,800,317	28.06	9,204,000	17.44
TOTAL	3,868,310	$113,137,191	$ 29.25	$ 49,139,073	$ 12.70

Tables of Cost Totals - Contractor

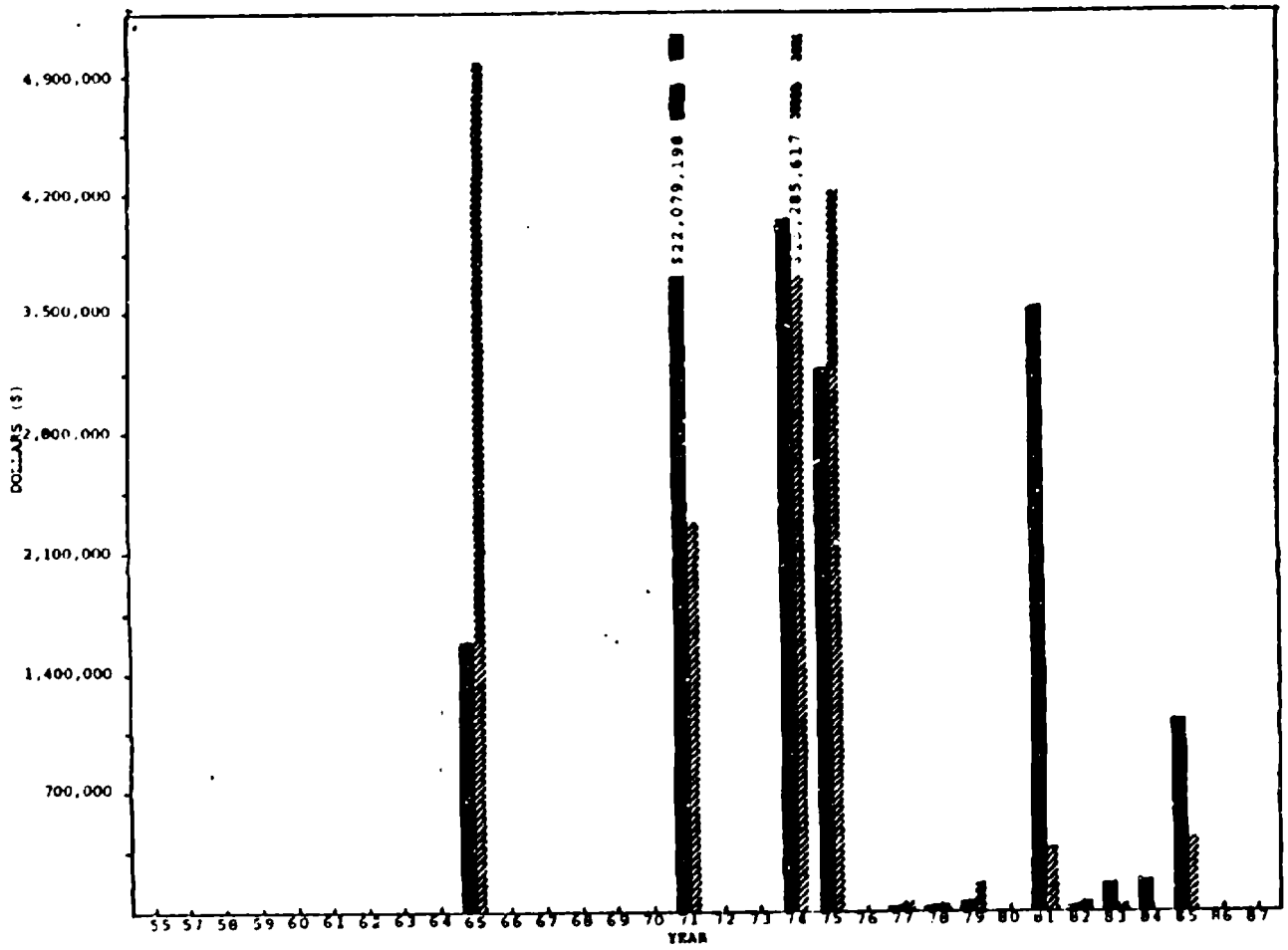
Appendix F: Graphs of Total Software Costs

Appendix F contains the charts comparing the development and maintenance cost per year over time for total AF, the four language categories, selected AF commands, and contractor developed software.

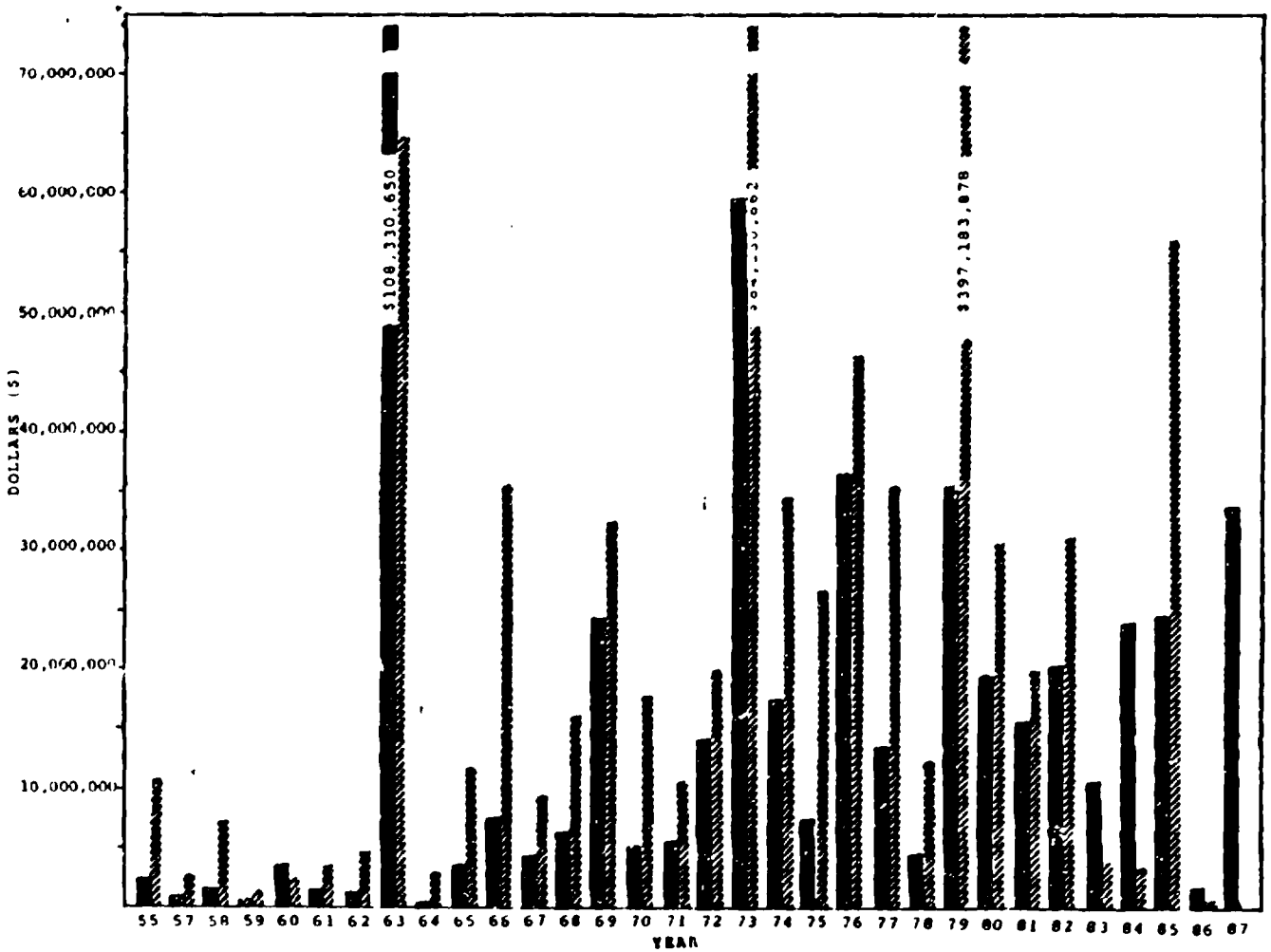
On the chart, development cost is represented by a solid bar. Maintenance cost is represented by a diagonally slashed bar. This legend is used on all the charts contained in this appendix.



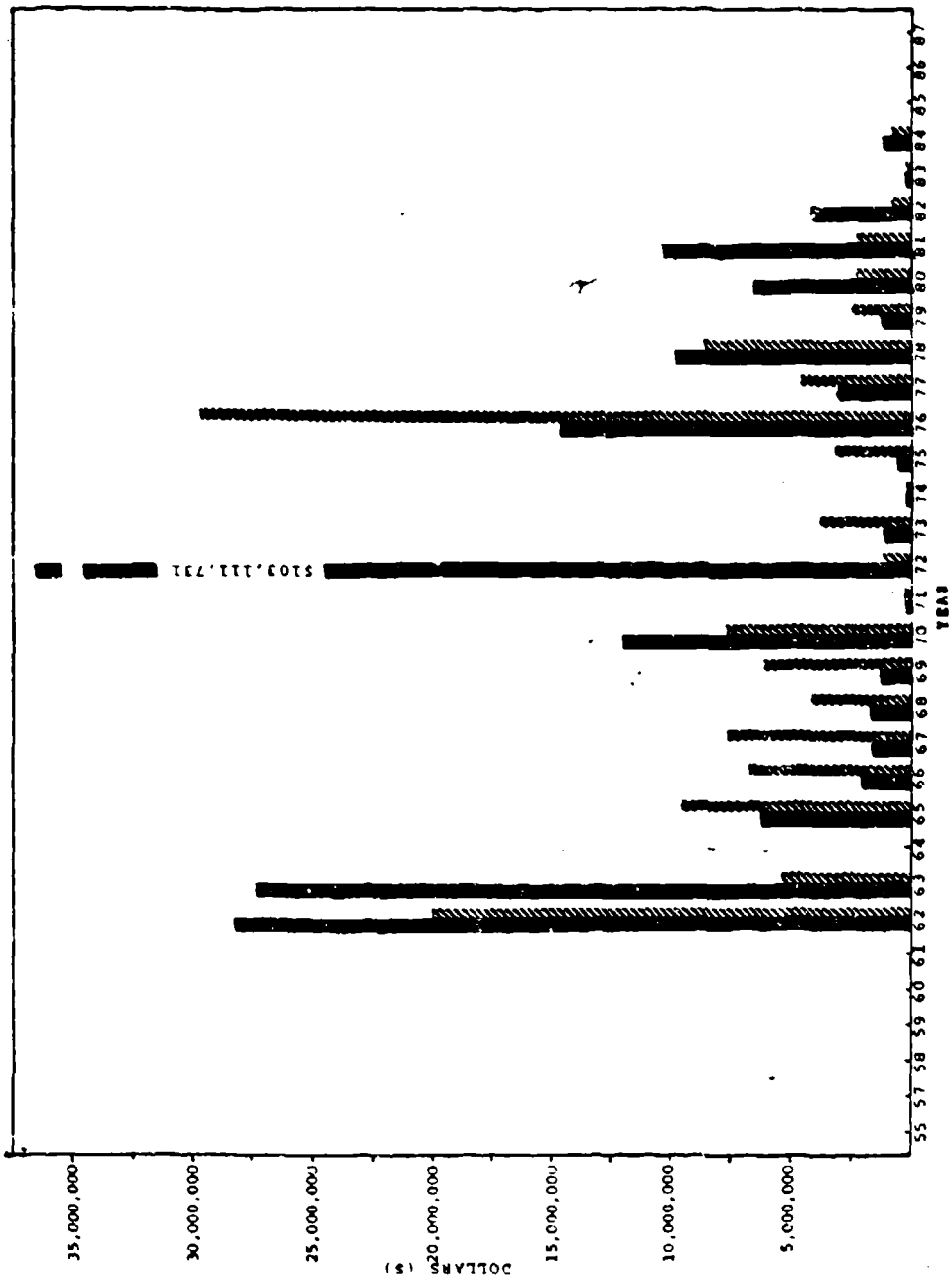
Graphs of Total Software Costs - Total AF



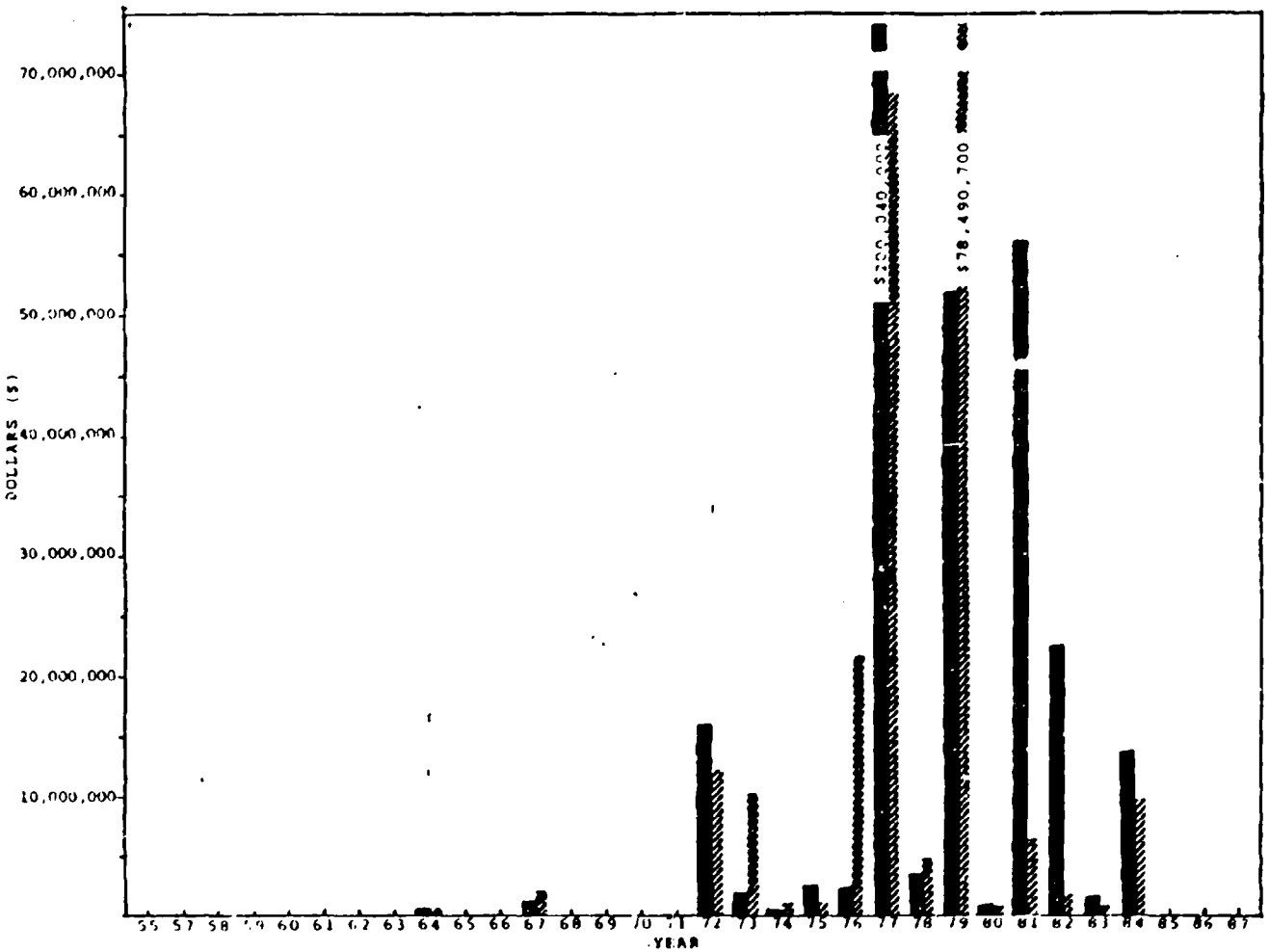
Graphs of Total Software Costs -- Languages - Assembler



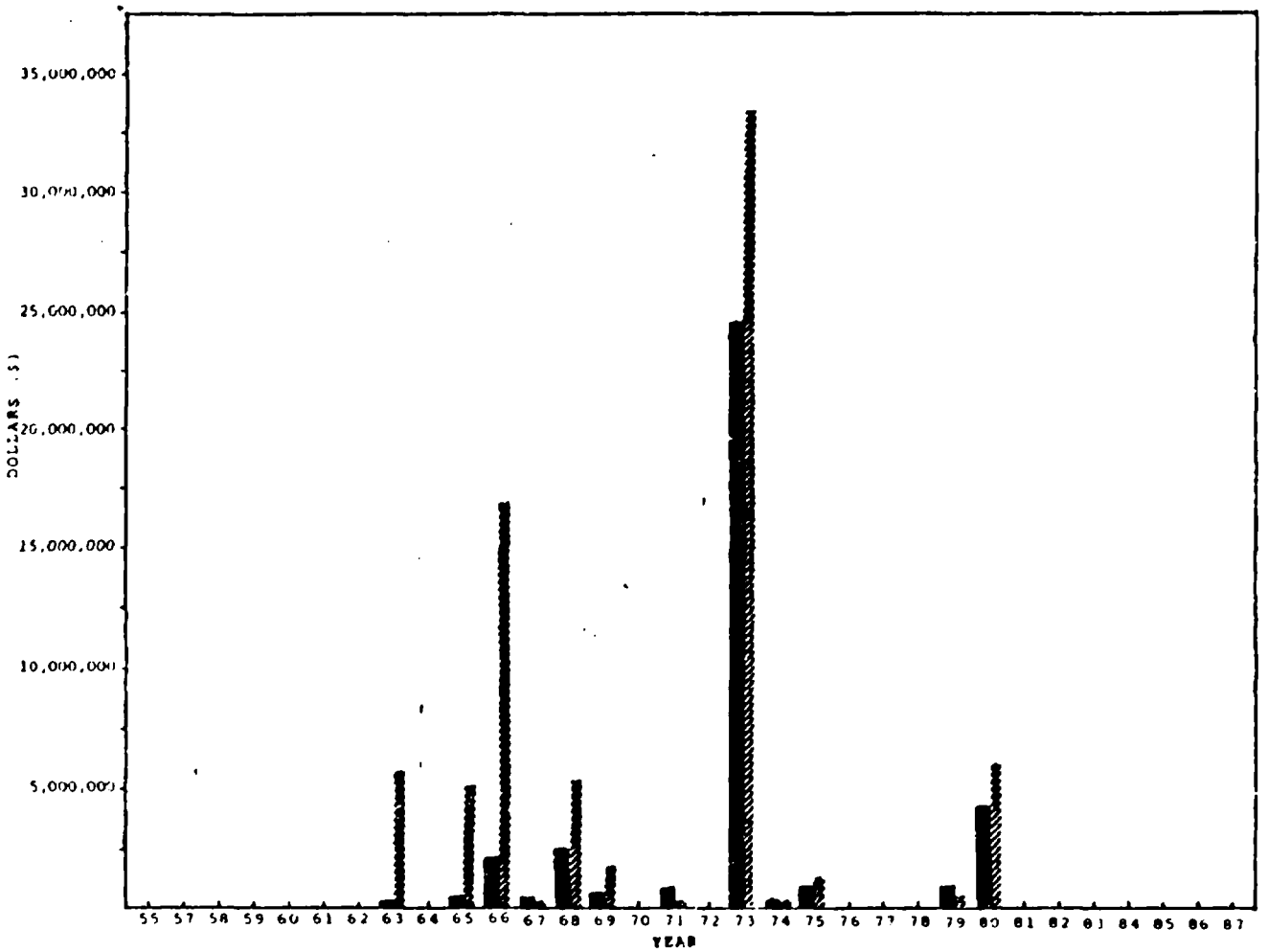
Graphs of Total Software Costs - Languages - COBOL



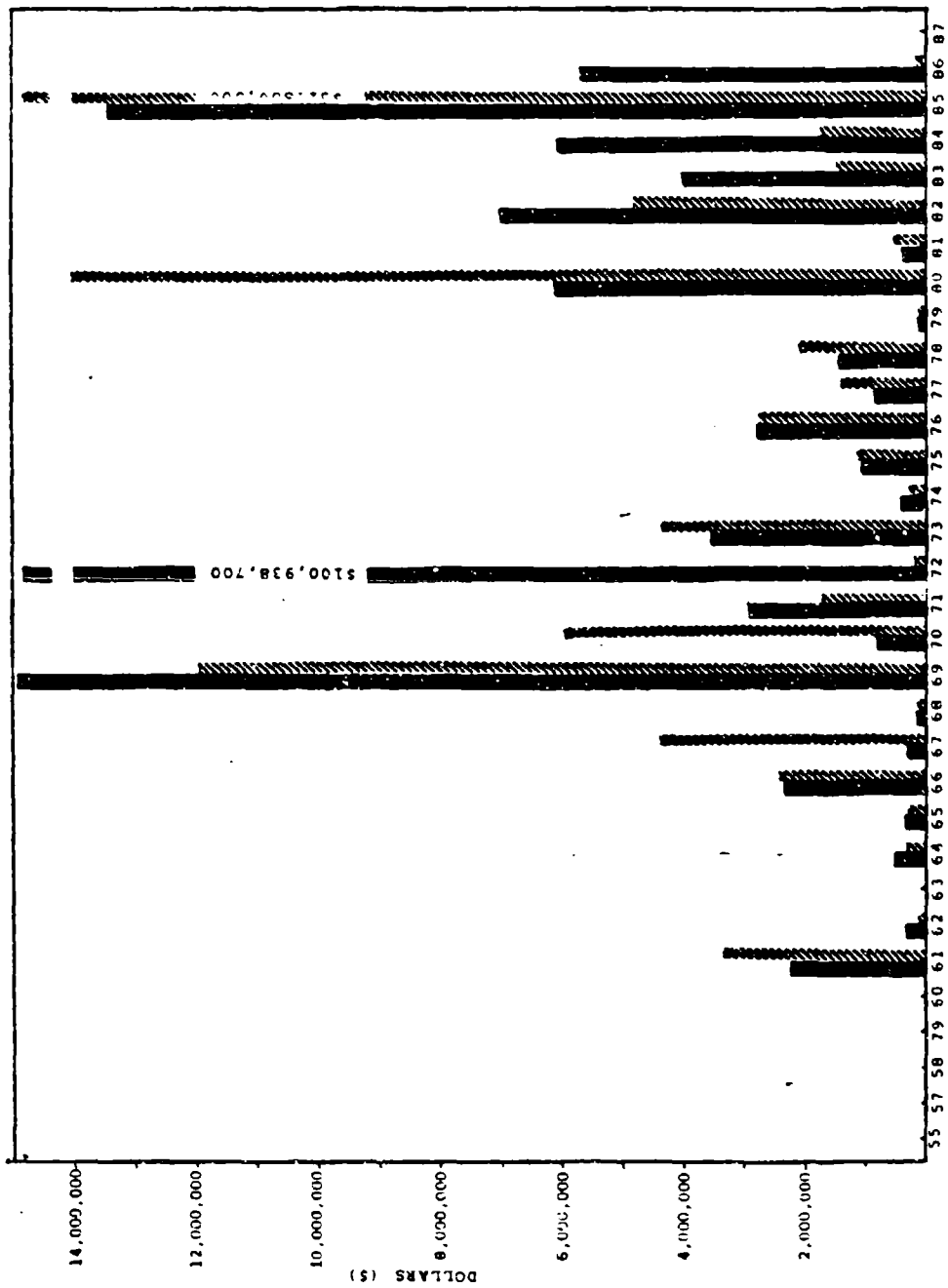
Graphs of Total Software Costs - Languages - FORTRAN



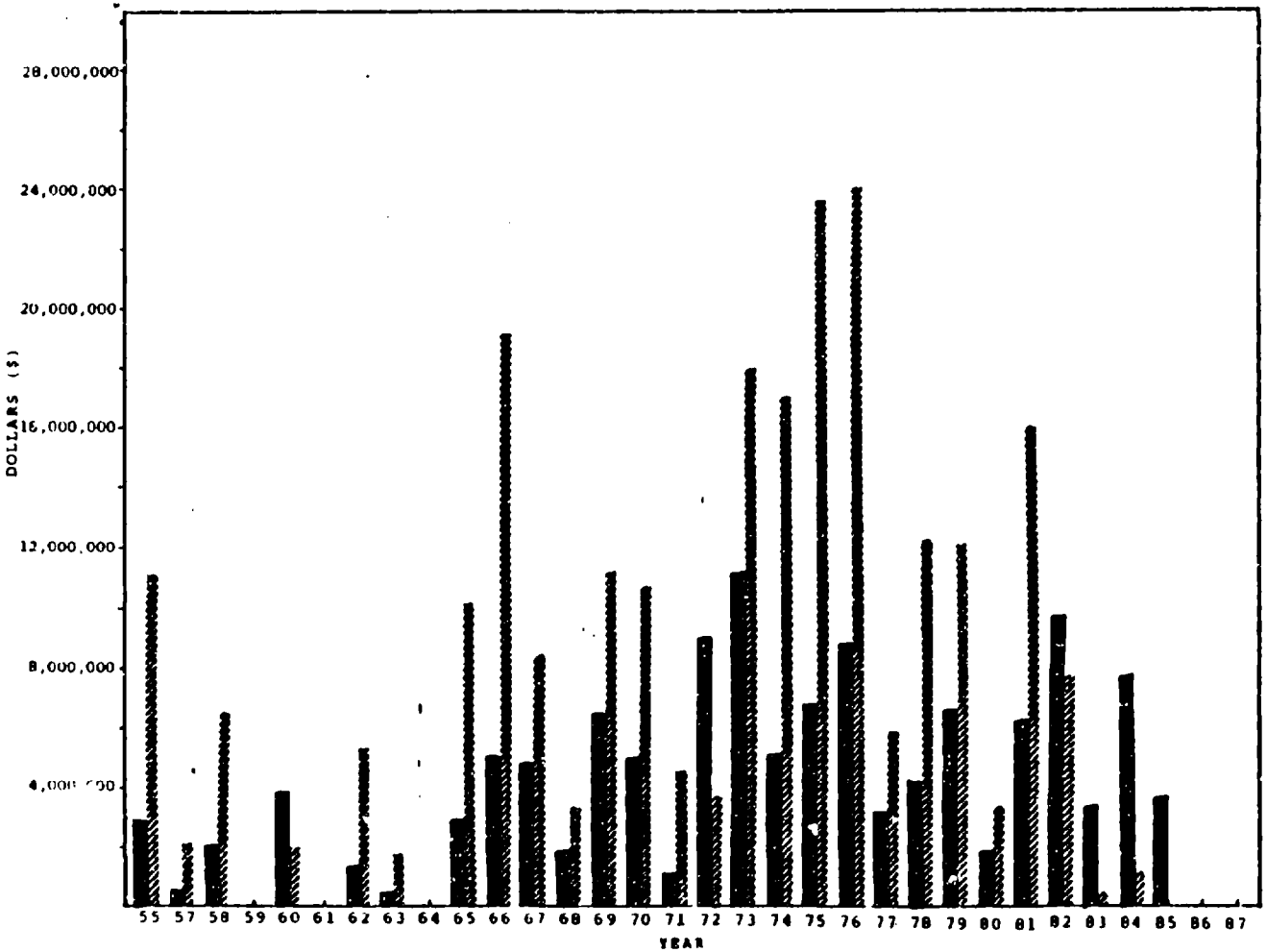
Graphs of Total Software Costs - Language - Other



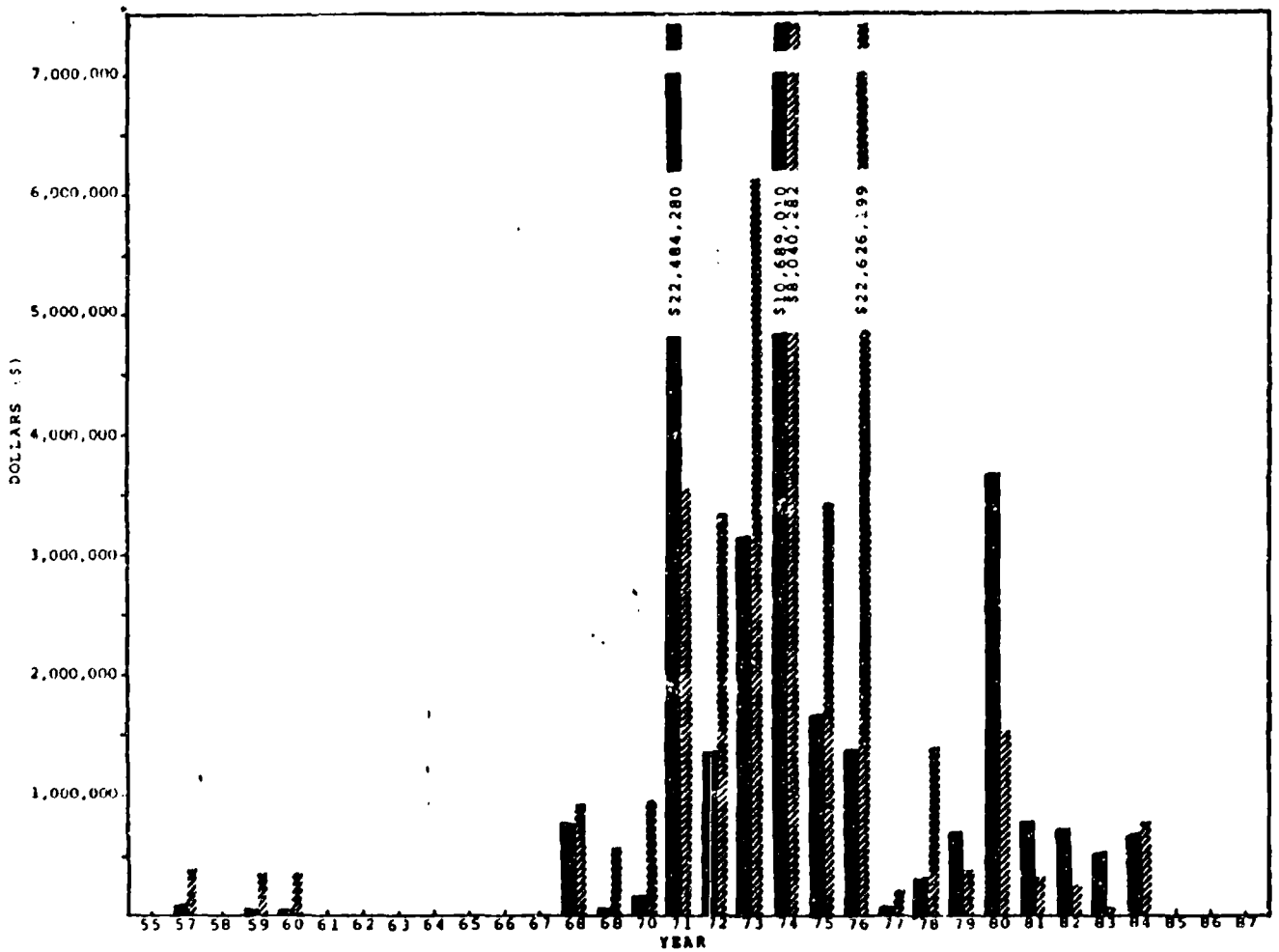
Graphs of Total Software Costs - Command - AFAFC



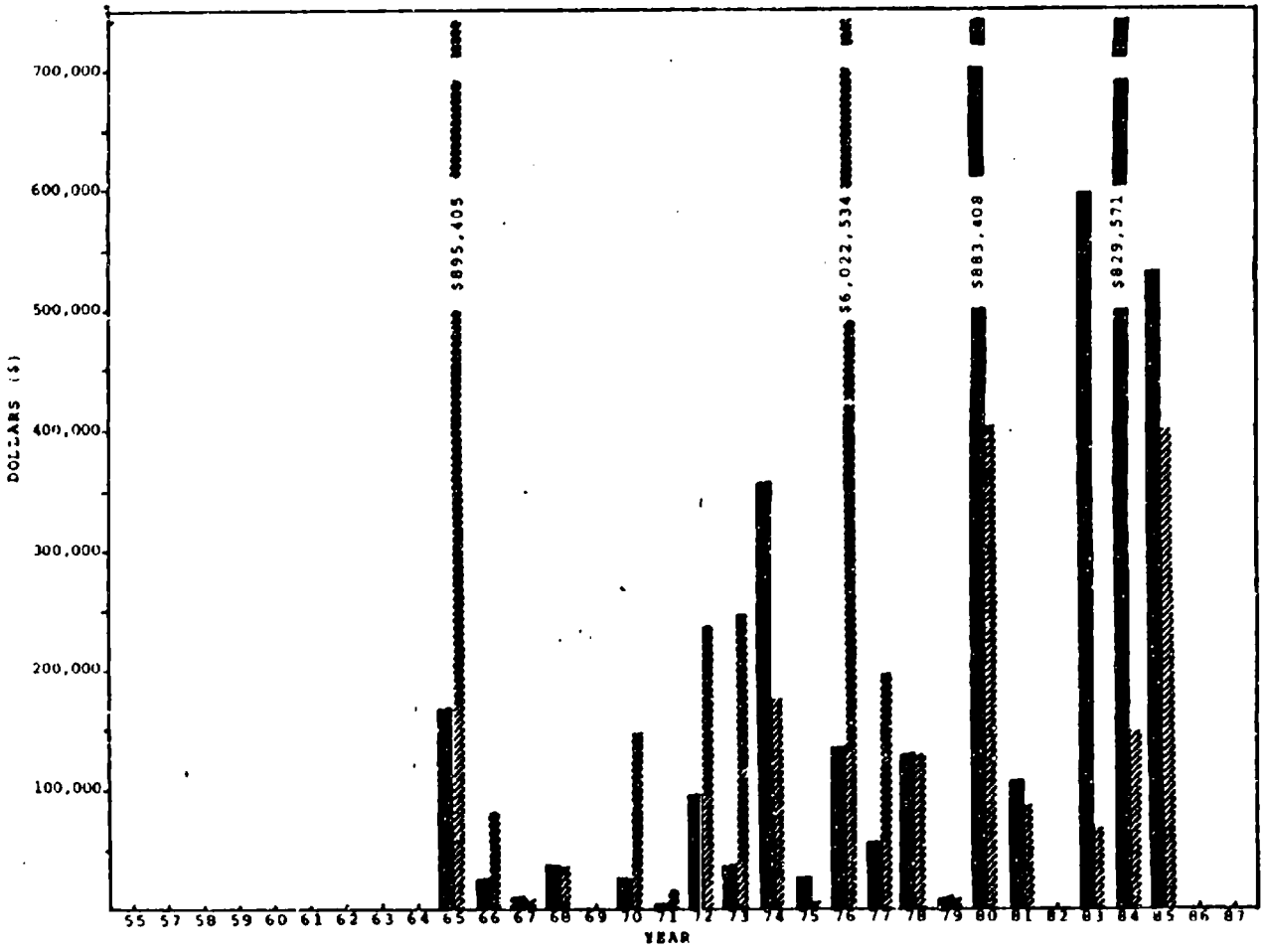
Graphs of Total Software Costs - Command - AFCC



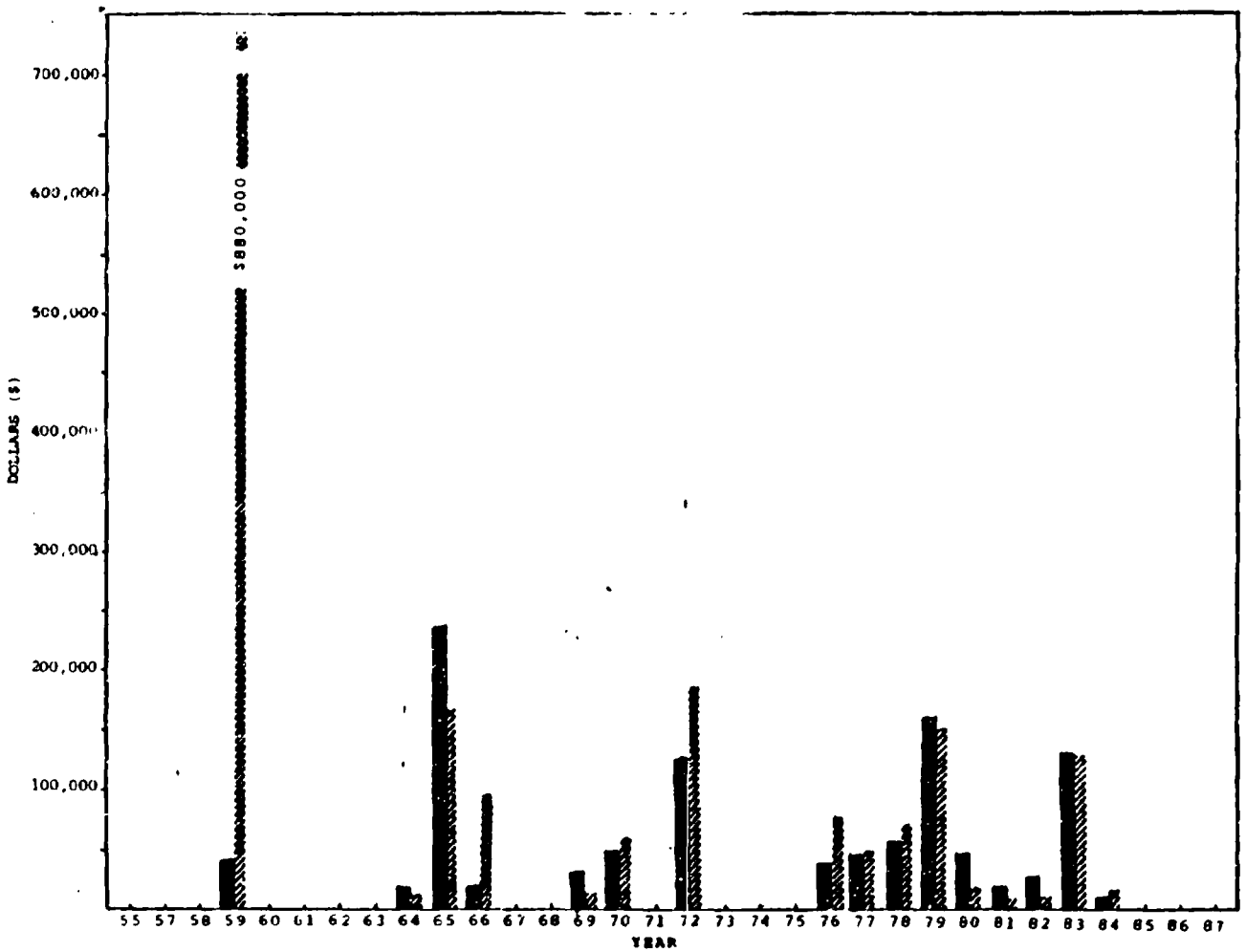
Graphs of Total Software Costs - AF Commands - AFLC



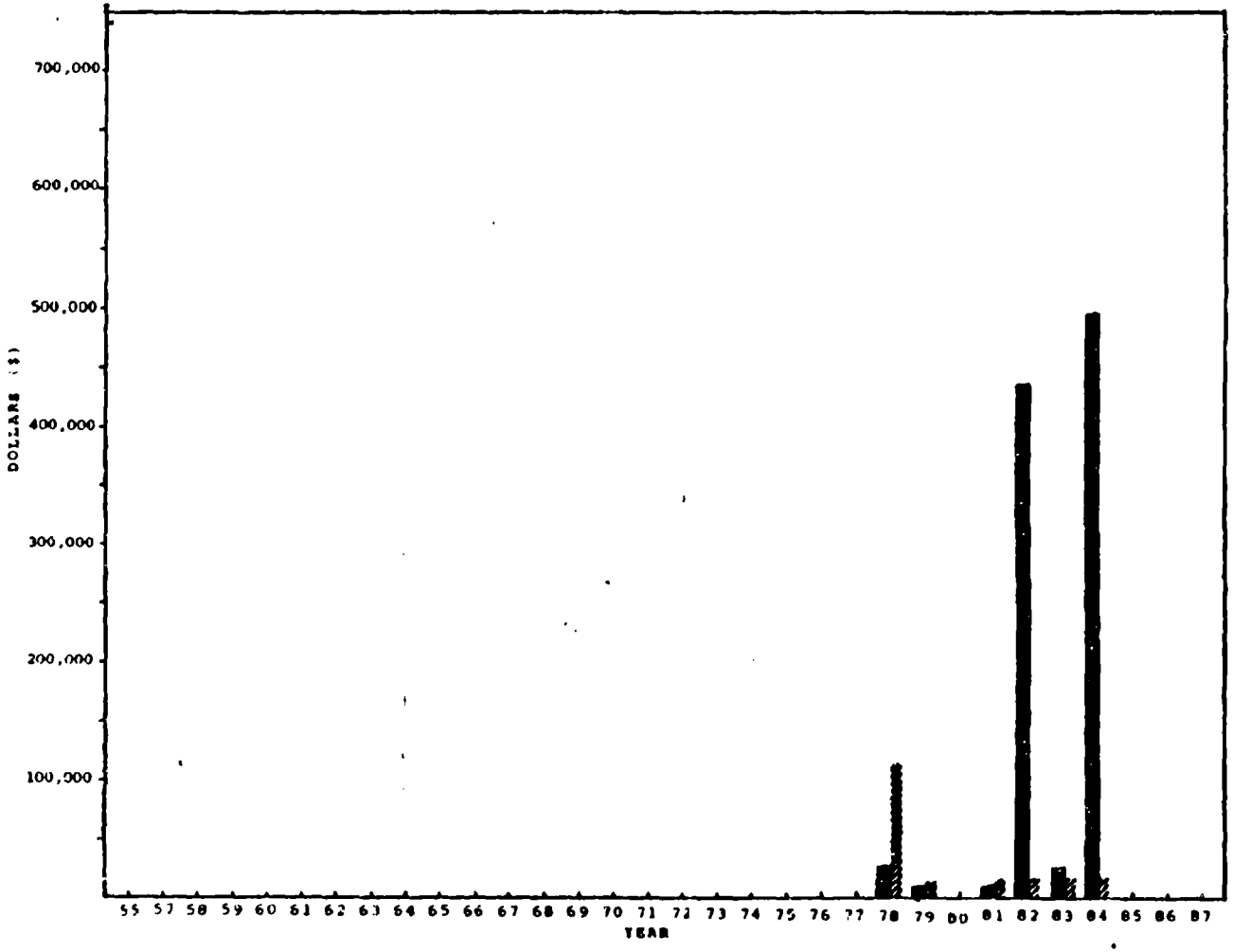
Graphs of Total Software Costs - AF Commands - AFSC



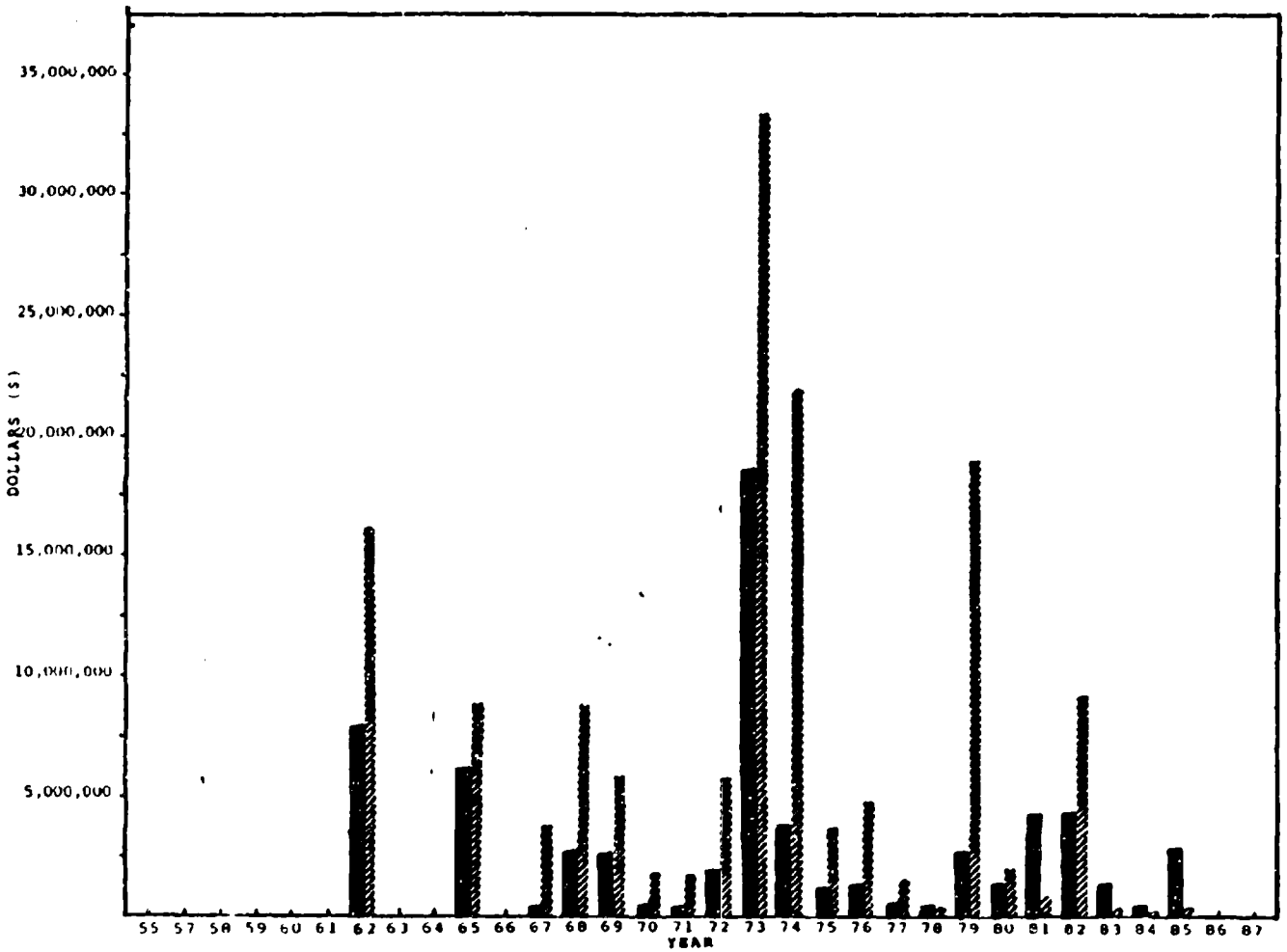
Graphs of Total Software Costs - AF Commands - ATC



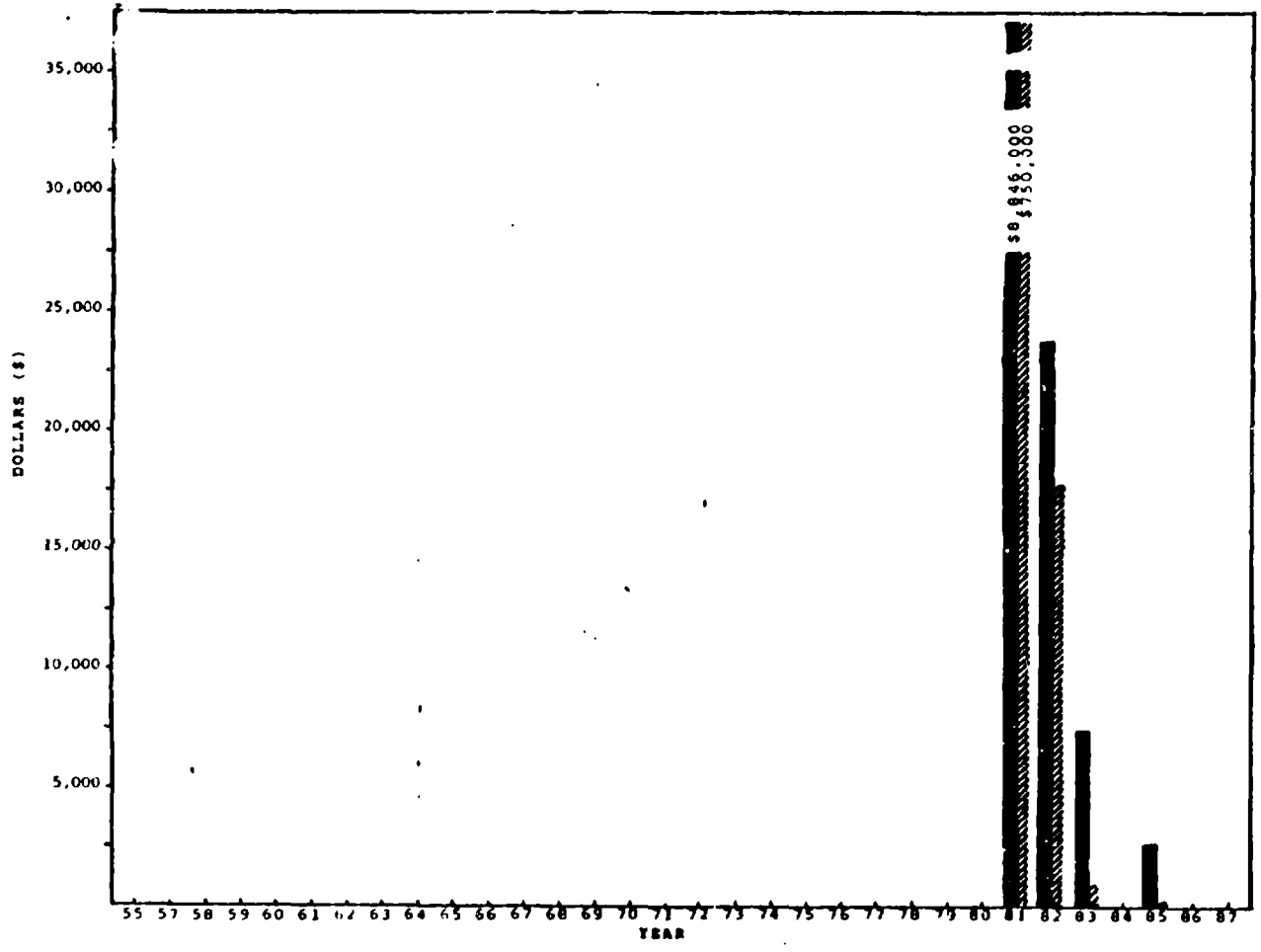
Graphs of Total Software Costs - AF Commands - AU



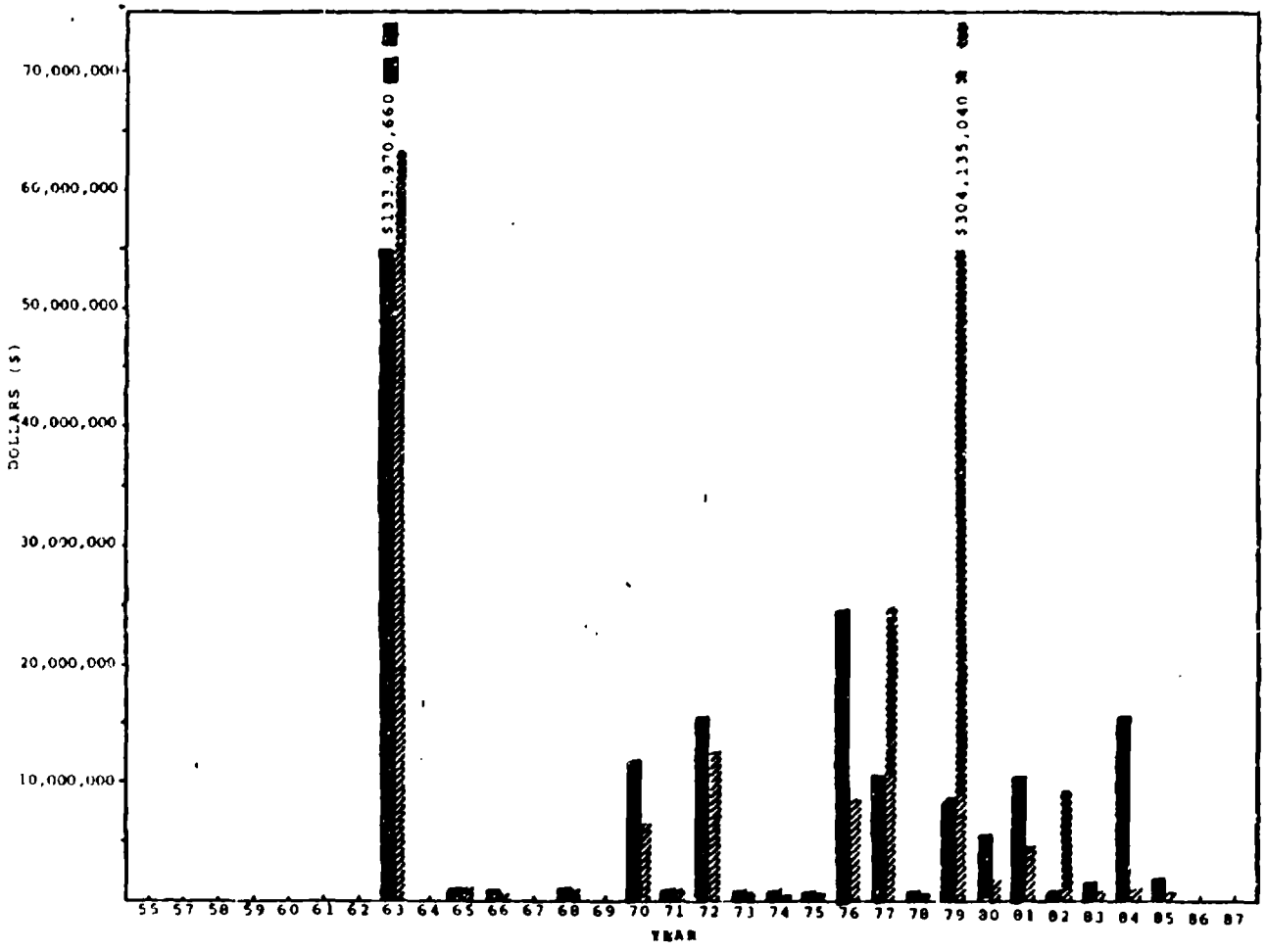
Graphs of Total Software Costs - AF Commands - ESC



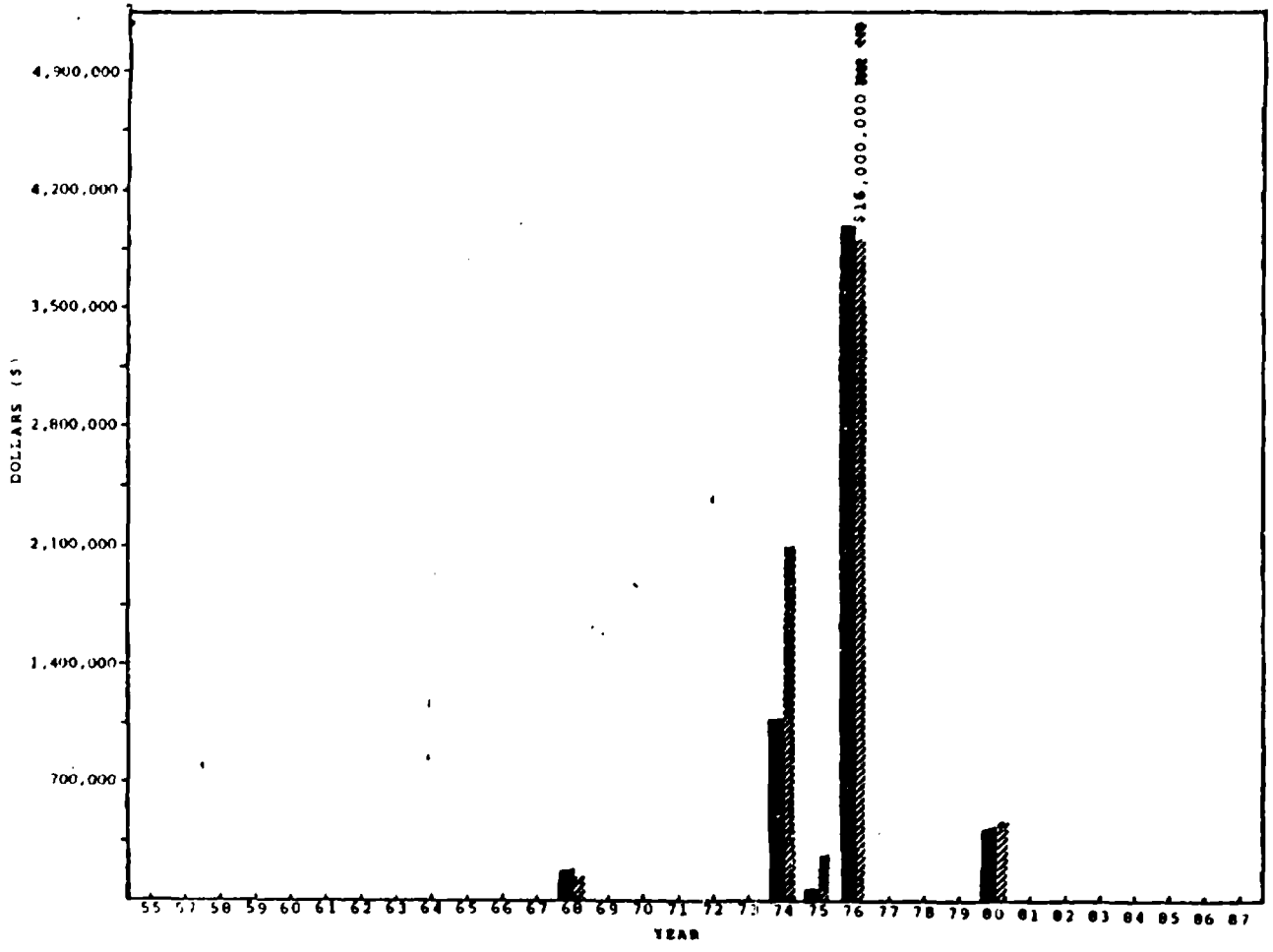
Graphs of Total Software Costs - AF Commands - MAC



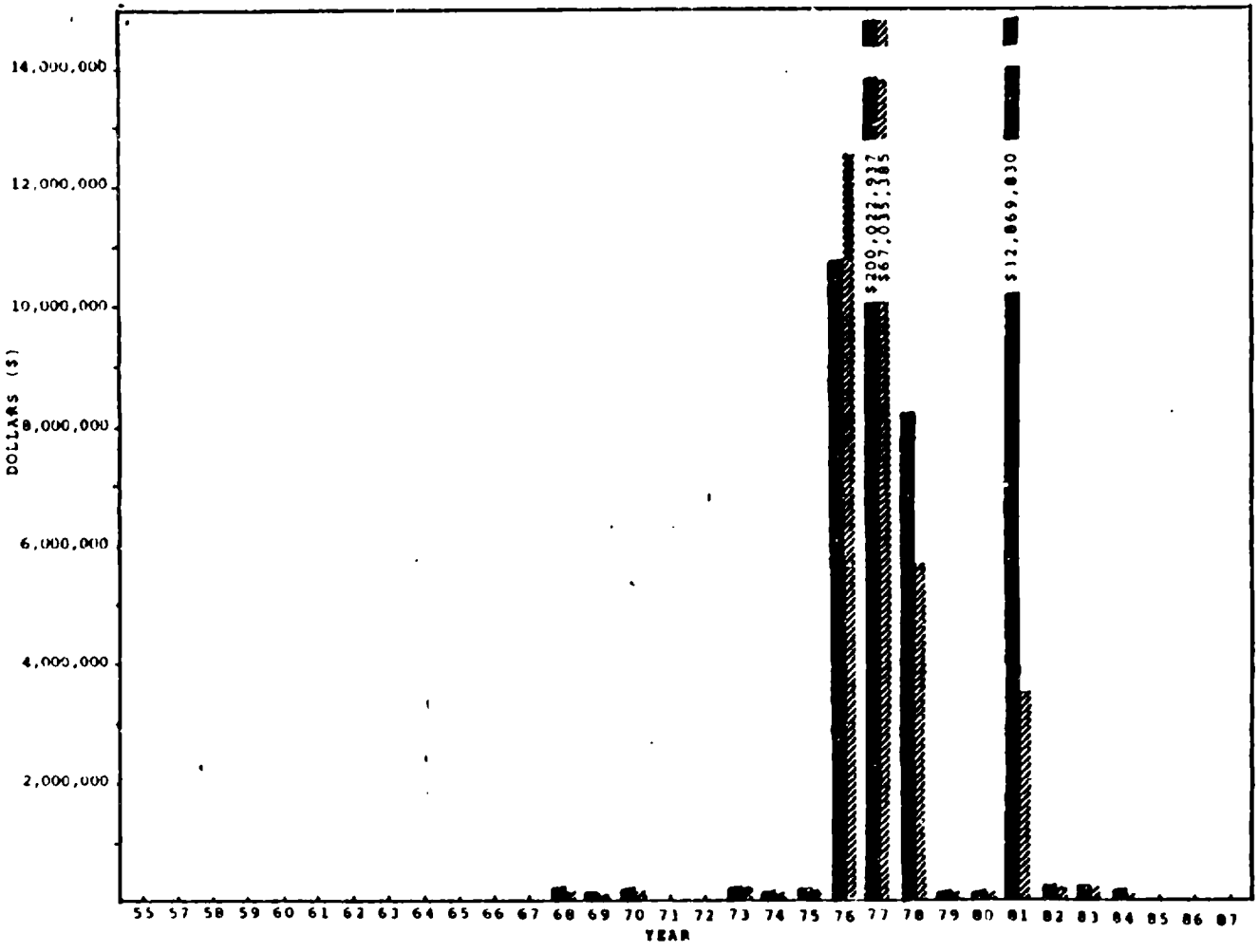
Graphs of Total Software Costs - AF Commands - PACAF



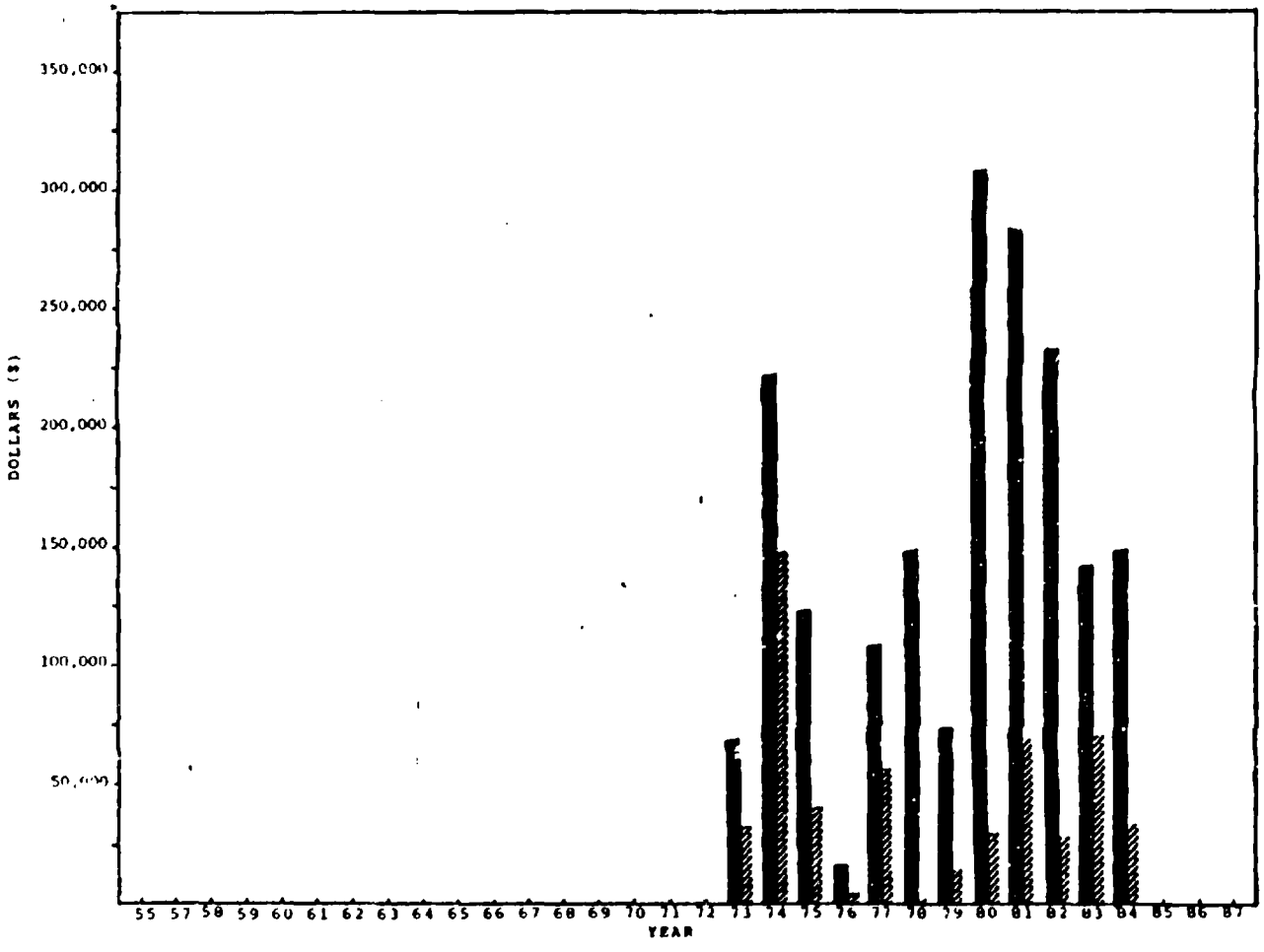
Graphs of Total Software Costs - AF Commands - SAC



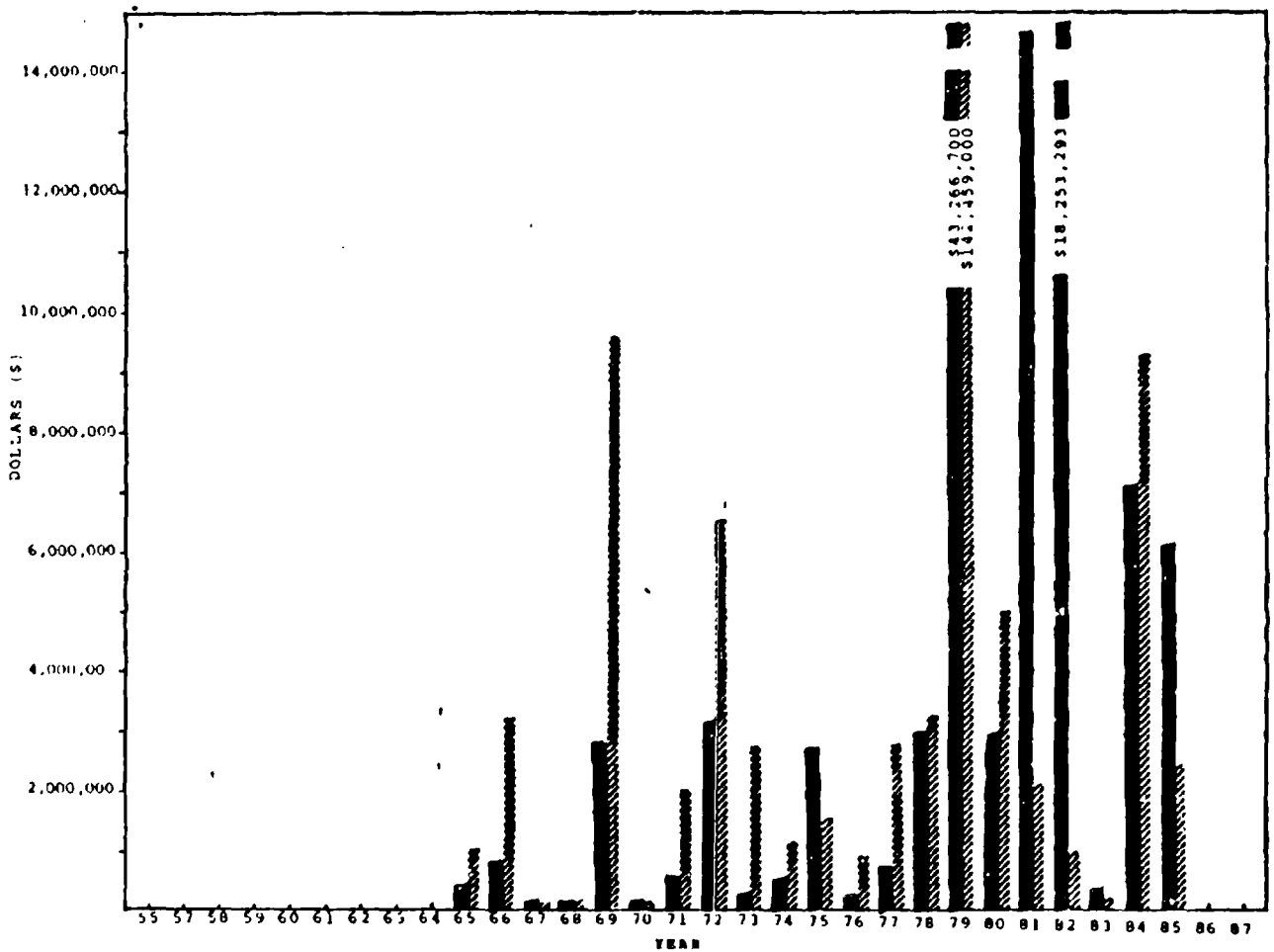
Graphs of Total Software Costs - AF Commands - SISC



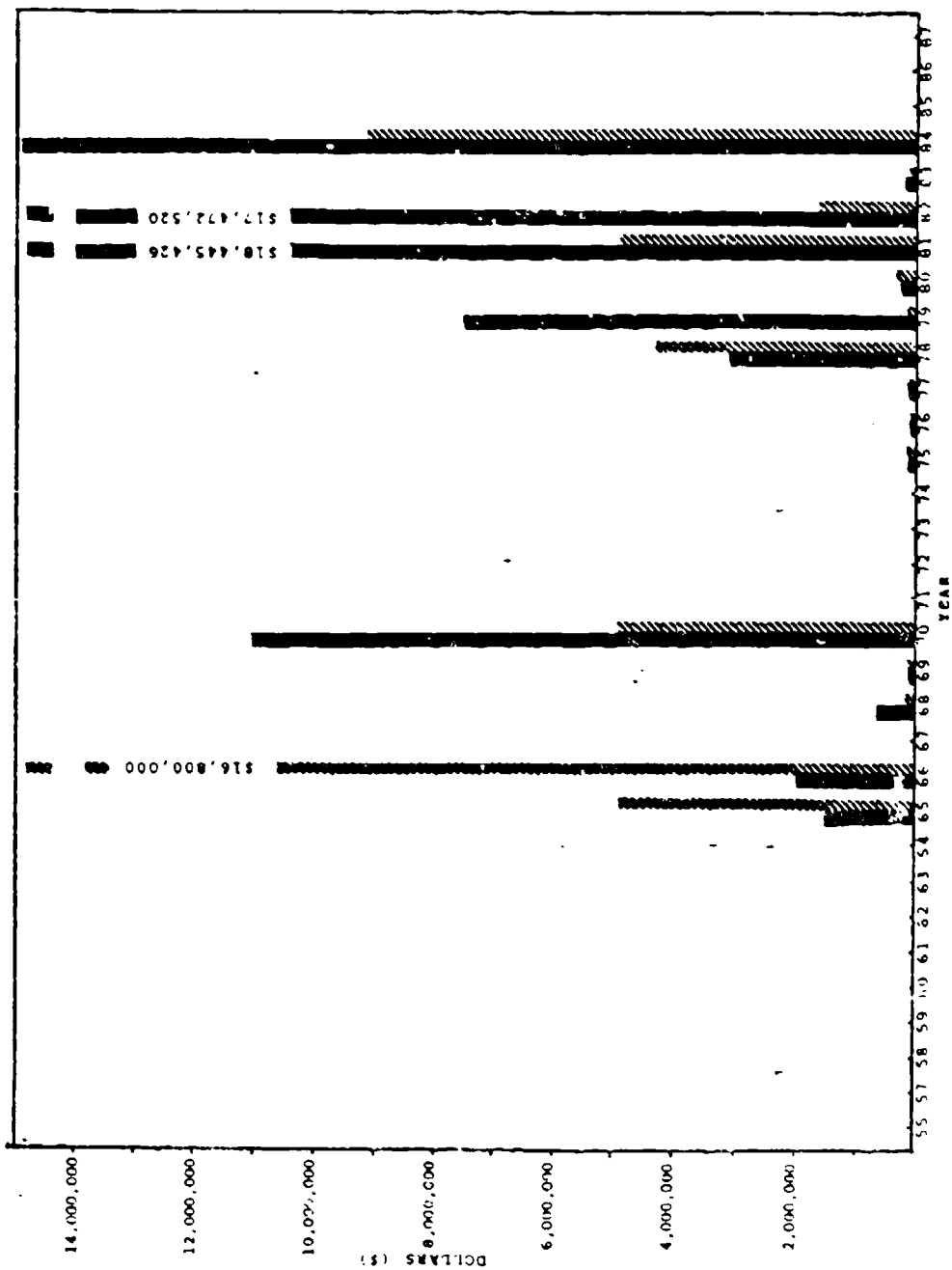
Graphs of Total Software Costs - AF Commands - TAC



Graphs of Total Software Costs - AF Commands - USAFE



Graphs of Total Software Costs - AF Commands - Others



Graphs of Total Software Costs - Contractor

Appendix G: Software Quality Assurance Factor Tradeoffs

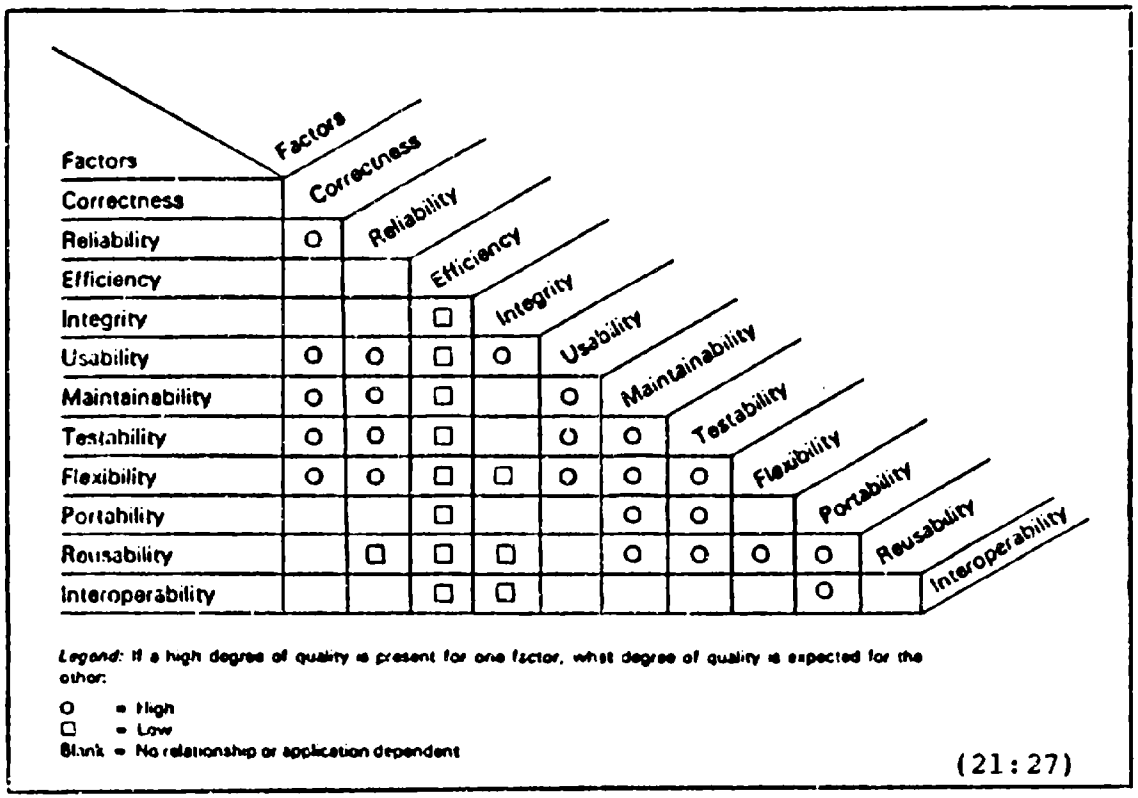
Factor	Definition
Correctness	Extent to which a program satisfies its specifications and fulfills the user's mission objectives.
Reliability	Extent to which a program can be expected to perform its intended function with required precision.
Efficiency	The amount of computing resources and code required by a program to perform a function.
Integrity	Extent to which access to software or data by unauthorized persons can be controlled.
Usability	Effort required to learn, operate, prepare input, and interpret output of a program.
Maintainability	Effort required to locate and fix an error in an operational program.
Testability	Effort required to test a program to ensure it performs its intended function.
Flexibility	Effort required to modify an operational program.
Portability	Effort required to transfer a program from one hardware configuration and/or software system environment to another.
Reusability	Extent to which a program can be used in other applications--related to the packaging and scope of the functions that the programs perform.
Interoperability	Effort required to couple one system with another (21:22).

Criterion	*Definition*	*Related Factors*
Traceability	Those attributes of the software that provide a thread from the requirements to the implementation with respect to the specific development and operational environment.	Correctness
Completeness	Those attributes of the software that provide full implementation of the functions required.	Correctness
Consistency	Those attributes of the software that provide uniform design and implementation techniques and notation.	Correctness Reliability Maintainability
Accuracy	Those attributes of the software that provide the required precision in calculations and outputs.	Reliability
Error Tolerance	Those attributes of the software that provide continuity of operation under non-nominal conditions.	Reliability
Simplicity	Those attributes of the software that provide implementation of functions in the most understandable manner. (Usually avoidance of practices which increase complexity.)	Reliability Maintainability Testability
Modularity	Those attributes of the software that provide a structure of highly independent modules.	Maintainability Flexibility Testability Portability Reusability Interoperability
Generality	Those attributes of the software that provide breadth to the functions performed.	Flexibility Reusability
Expandability	Those attributes of the software that provide for expansion of data storage requirements or computational functions.	Flexibility
Instrumentation	Those attributes of the software that provide for the measurements of usage or identification of errors.	Testability
Self-Descriptiveness	Those attributes of the software that provide explanation of the implementation of a function.	Flexibility Maintainability Testability Portability Reusability

(21:25)

Criterion	*Definition*	*Related Factors*
Execution Efficiency	Those attributes of the software that provide for minimum processing time.	Efficiency
Storage Efficiency	Those attributes of the software that provide for minimum storage requirements during operation.	Efficiency
Access Control	Those attributes of the software that provide for control of the access of software and data.	Integrity
Access Audit	Those attributes of the software that provide for an audit of the access of software and data.	Integrity
Operability	Those attributes of the software that determine operation and procedures concerned with the operation of the software.	Usability
Training	Those attributes of the software that provide transition from current operation or initial familiarization.	Usability
Communicativeness	Those attributes of the software that provide useful inputs and outputs which can be assimilated.	Usability
Software System Independence	Those attributes of the software that determine its dependency on the software environment (operating systems, utilities, input/output routines, etc.).	Portability Reusability
Machine Independence	Those attributes of the software that determine its dependency on the hardware system.	Portability Reusability
Communications Commonality	Those attributes of the software that provide the use of standard protocols and interface routines.	Interoperability
Data Commonality	Those attributes of the software that provide the use of standard data representations.	Interoperability
Conciseness	Those attributes of the software that provide for implementation of a function with a minimum amount of code.	Maintainability

(21:26)



Factors	Correctness	Reliability	Efficiency	Integrity	Usability	Maintainability	Testability	Flexibility	Portability	Reusability	Interoperability
Correctness											
Reliability	O										
Efficiency											
Integrity			□								
Usability	O	O	□	O							
Maintainability	O	O	□		O						
Testability	O	O	□		O	O					
Flexibility	O	O	□	□	O	O	O				
Portability			□			O	O				
Reusability		□	□	□		O	O	O	O		
Interoperability			□	□					O		

Legend: If a high degree of quality is present for one factor, what degree of quality is expected for the other:

O = High
□ = Low
Blank = No relationship or application dependent

(21:27)

Appendix H: Definitions of Maintenance Activities

Requirements Analysis

Evaluation of the impact of a change in requirements or the reason why a current requirement is not satisfied. Determination and assimilation of the current software documentation necessary to understand the nature of the change required.

Design Analysis

Evaluation of the impact of a modification and development of a strategy of redesign. A decision typical of this activity is whether a portion of the system has to be redesigned or whether the modification can be made within the context of the current software architecture. Also included is an evaluation of modification to the data base design.

User Interface

Activities associated with interacting with the user/customer. This activity may involve formal documentation, meetings, and walkthroughs, etc.

Design Review

Formal or informal review of the design analysis activity.

Problem Report Recording and Control

Includes all activities associated with how users report problems, how problems are logged, assigned priority, response time, and closed.

Configuration Control

Includes all activities associated with maintaining baseline version of code.

Data Base Modification

Modification made to data base structure and individual data values.

Code Modification/Recompilation

Changes made to code by programmers to repair an error or to enhance the operation of the system.

Code Debugging/Module Testing

Includes testing after code changes have been made and investigative debugging, i.e. testing to identify where an error source is.

Subsystem Testing

Testing groups of modules or programs to assess whether modifications have been made correctly.

System Testing

Tests run to determine if new version of software, due to corrective, perfective, or adaptive maintenance, operates correctly. Typically called regression testing if some set of test cases/test data used. Acceptable completion of these tests is usually the basis for fielding the new version of the system.

Documentation Modification

Activities including changes to system specifications, users manuals, maintenance manuals, etc., made as a result of modification to system.

Standards Audit

Activities performed to insure new version of system and documentation meet established standards prior to fielding.

Code Inspections/Walkthroughs

Review of modifications to code.

Test Data Generation

Development of test data to verify and validate code changes.

Management Planning and Control

Management activities related to planning, control, personnel assignment, prioritization of jobs, personnel requirements estimation, budget estimation and control, scheduling, etc.

Field Delivery

Activities associated with fielding updated system.

Software Support Development/Maintenance

Development and maintenance of tools used in supporting above activities.

Hardware Support

Procurement and maintenance of hardware system used as maintenance facility.

Administrative Support

Secretarial, data entry, and clerk support to maintenance personnel (19:9-12).

Appendix I: Metrics of Maintenance

Basic Measure of Maintenance Activities:

1. Number of Lines of Code/Number Modified.
2. Number of Data Items/Number Modified.
3. Number of Modules/Number Modified/Number Added.
4. Number of Functions/Number Modified/Number Added.
5. Number of Interfaces/Number Modified/Number Added.
6. Number of Requirements/Structure Changes/Number Changed/Number Added/Links Changed.

Measures of Maintainability of System:

1. Traceability--Measure of traceability from requirements to code.
2. Consistency--Measures of use of standard data definition, naming, documentation conventions and Requirements Consistency.
3. Conciseness--Halstead's Length/Effort Measures.
4. Modularity--Lines of code by module profile Called/call matrix.
5. Self-description--Number of comments/LOC.
6. Stability--Myer's Stability Measure.
7. Effort/Cost--CPU run time, Average time to fix.
8. Complexity--McCabe's Cyclomatic Number.

Measures of Reliability of System:

1. Number of User Problem Reports.
2. Number of User Problem Reports per line of code.
3. Number of User Problem Reports induced as a result of a maintenance activity.
4. Number of User Problem Reports induced as a result of a maintenance activity per line of code modified.
5. System Reliability (MTBF).
6. System Availability $\left(\frac{\text{MTBF - MTTR}}{\text{MTBF}}\right)$.
7. Completeness of Requirements.
8. Error Categorization/Impact Assessment (19:70).

Bibliography

1. Balzer, Robert and others. "Software Technology in the 1990's: Using a New Paradigm," Computer, 16: 39-45 (November 1985).

2. Bartol, Kathryn M. "Turnover Among DP Personnel: A Causal Analysis," Communications of the ACM, 26: 807-811 (October 1983).

3. Berard, Edward V. "Ada Education--A Moving Target," Defense Science & Electronics, 3: 51-55 (May 1984).

4. Berney, Karen. "Aging Software Swamps Bureaucracy," Electronics, 57: 115-116 (June 14, 1984).

5. Boehm, Barry. Software Engineering Economics. Englewood Cliffs NJ: Prentice-Hall, Inc., 1981.

6. Brice, Linda and John Connell. "A Methodology for Minimizing Maintenance Costs," DATAPRO, AS75-100: 151-158 (September 1984).

7. Buss, Martin D. J. "Guides to Maintaining Key Applications Programs," DATAPRO, AS75-060: 101-107 (April 1982).

8. Butler, Charles W. and others. "Mending Crazy Quilt Systems," Datamation, 30: 130-142 (May 15, 1984).

9. Canan, James W. "The Software Crisis," Air Force Magazine, 69: 46-52 (May 1986).

10. Childress, Maj Robert E., Jr. Contractor versus Organic Maintenance for Space Command Automatic Data Processing Equipment. MS thesis, AFIT/GLM/LSM/85S-13. School of Systems and Logistics, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, September 1985 (AD-A162307).

11. Connell, John and Linda Brice. "Prolonging the Life of Software," DATAPRO, AS75-300: 101-107 (May 1985).

12. Department of the Air Force. Manpower: Manpower Policies and Procedures Comparative Cost Analysis. AFR 26-1. Washington: HQ USAF, 2 October 1981.

13. Dunn, Robert and Robert Ullman. Quality Assurance for Computer Software. New York: McGraw-Hill Book Company, 1982.

14. Er, M. C. "Principles of Program Documentation," DATAPRO, AS75-120: 101-105 (November 1984).

15. Ferens, Daniel V. Mission Critical Computer Software Support Management. Wright-Patterson AFB OH: Air Force Institute of Technology, School of Systems and Logistics, Department of Systems Acquisition Management, February 1986.

16. Green, Lt Cmdr James F. and Lt Brena F. Selby. Dynamic Planning and Control of Software Maintenance: A Fiscal Approach. MS thesis. Naval Postgraduate School, Monterey CA, December 1981 (AD-A112801).

17. Guimaraes, Tur. "Managing Application Program Maintenance Expenditures," Communications of the ACM, 26: 739-746 (October 1983).

18. Henderson, Brian J. and Brenda Sullivan. "How to Estimate Software Maintenance Costs," DATAPRO, AS75-105: 101-113 (February 1986).

19. Herndon, Mary Anne and Jim McCall. Software Maintenance Guidelines. Seminar for National Bureau of Standards, Institute for Computer Sciences and Technology (Contract No. NB825BCA1647). Science Applications, Inc., La Jolla CA.

20. Joyce, Capt James P. A Study of the Software Maintenance Process of Air Force Weapon Systems. MS thesis, AFIT/GCS/MA/82D-5. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, December 1982 (AD-A124758).

21. Lamb, Capt Steven P. A Survey and Evaluation of Software Quality Assurance. MS thesis, AFIT/GSM/LSY/84S-19. School of Systems and Logistics, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, September 1984 (AD-A147552).

22. Lientz, Bennet P. and E. Burton Swanson. "Problems in Application Software Maintenance," DATAPRO, AS75-050: 101-108 (December 1982).

23. Martin, Roger J. and Wilma M. Osborne. U.S. Department of Commerce, National Bureau of Standards. Guidance on Software Maintenance. NBS Special Publication 500-106. Washington: Government Printing Office, 1983.

24. McCall, James A. and others. U.S. Department of Commerce, National Bureau of Standards. Software Maintenance Management. NBS Special Publication 500-129. Washington: Government Printing Office, 1985.

25. McClure, Carma L. "Designing Software with Maintenance in Mind," DATAPRO, AS75-075: 101-117 (March 1982).

26. Osborne, Wilma M. Telephone interview. Center for Programming Science and Technology, Institute for Computer Sciences and Technology, National Bureau of Standards, Gaithersburg MD, 24 July 1986.

27. Perry, William E. Hatching the EDP Quality Assurance Function. Orlando FL: Quality Assurance Institute, 1981.

28. -----. "Time to Change Attitudes about Software Maintenance," Government Computer News, 5: 61 (June 6, 1986).

29. Peterson, Robert O. "Maintenance Isn't Maintenance Anymore," Data Processing Digest, 30: 5-6 (August 1984).

30. Schatz, Willie. "Fed Facts," Datamation, 32: 72 (15 Aug 1986).

31. Tayntor, Christine B. "Goal Setting in the Maintenance Department," Data Management, 24: 36 (February 1986).

32. Thompson, Gene E. "Underworked and Overlooked: User Documentation," Data Processing Digest, 29: 13 (June 1983).

33. Yourdon, Edward. Techniques [illegible]rogram Structure and Design. Englewood Cliffs NJ: Prentice-Hall, Inc., 1975.

Vita

Captain Robert E. NeSmith was born on 14 April 1954 in Hawkinsville, Georgia. He graduated from high school in Enterprise, Alabama in 1972. He received the degree of Bachelor of Science in Computer Science from Troy State University in June 1978. Upon graduation, he received his commission through Air Force ROTC. From October 1978 to July 1982, he was assigned to the Computer Support DCS at HQ AFLC at Wright-Patterson AFB, Ohio. He served as an application programmer, computer operations officer, and executive officer. From August 1982 to May 1985, he served as a computer staff officer at HQ SAC in the computer support DCS. In May 1985, he entered the Air Force Institute of Technology. Captain NeSmith will receive the degree of Master of Science in Systems Management from AFIT in September 1986.

Permanent address: 201 Forest Drive
Bellevue, Nebraska 68005

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

ADA174454

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION: UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT: Approved for public release; distribution unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S): AFIT/GSM/LSM/86S-15	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION: School of Systems and Logistics; 6b. OFFICE SYMBOL (If applicable): AFIT/LSM	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State and ZIP Code): Air Force Institute of Technology, Wright-Patterson AFB, Ohio 45433-6583	7b. ADDRESS (City, State and ZIP Code)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION; 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State and ZIP Code)	10. SOURCE OF FUNDING NOS. — PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT NO.
11. TITLE (Include Security Classification): See Box 19	

12. PERSONAL AUTHOR(S): Robert E. NeSmith II, B.S., Captain, USAF

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Yr., Mo., Day)	15. PAGE COUNT
MS Thesis	FROM _____ TO _____	1986 September	163

16. SUPPLEMENTARY NOTATION

17. COSATI CODES		
FIELD	GROUP	SUB. GR.
09	02	
05	02	

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number): Computers, Costs, Maintenance, Computer Programming, Programming

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

Title: A STUDY OF SOFTWARE MAINTENANCE COSTS OF AIR FORCE LARGE SCALE COMPUTER SYSTEMS

Thesis Chairman: James D. Meadows, GM-13, USAF
Associate Professor of Computer Systems Analysis

Approved for public release: IAW AFR 190-1.
LYNN E. WOLAVER 29 Sep 86
Dean for Research and Professional Development
Air Force Institute of Technology (AFIT)
Wright-Patterson AFB OH 45433

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION
UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS ☐	UNCLASSIFIED

22a. NAME OF RESPONSIBLE INDIVIDUAL	22b TELEPHONE NUMBER (Include Area Code)	22c. OFFICE SYMBOL
James D. Meadows, GM-13, USAF	(513) 255-4149	AFIT/LSM

DD FORM 1473, 83 APR — EDITION OF 1 JAN 73 IS OBSOLETE.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

Software maintenance is a growing concern throughout the software community. Due to the rising cost of computer software and the even greater increase in the software maintenance share of the budget, maintenance is becoming the major cost in a data processing organization.

This thesis examines the maintenance costs of Air Force "organic" software for the last thirty years. Generally, the cost per line of code and the total costs are rising for the large scale automatic data processing computer systems. Contractor developed software is also examined and its influence on Air Force software costs.